Noname manuscript No.
(will be inserted by the editor)

New subspace method for unconstrained black box optimization

Morteza Kimiaei[1] · **Arnold Neumaier** · **Parvaneh Faramarzi**

Received: date / Accepted: date

Abstract This paper defines an efficient subspace method, called **SSBBO**, for unconstrained black box optimization problems where the objective function has Lipschitz continuous gradients, but only function values contaminated by rounding errors are available. **SSBBO** employs line searches along directions constructed on the basis of quadratic models. These approximate the objective function in a subspace spanned by some of the previous search directions. A worst case complexity bound on the number of iterations and function evaluations is derived for a basic algorithm using this technique. Numerical results for a practical variant with additional heuristic features show that **SSBBO** has superior performance compared to the best solvers from the literature.

Keywords Unconstrained black box optimization · Subspace technique · Line search approach · Complexity results

Mathematics Subject Classification (2000) primary 90C56

1. The first author acknowledges the financial support of the Doctoral Program *Vienna Graduate School on Computational Optimization* (*VGSCO*) funded by the Austrian Science Foundation under Project No W1260-N35

M. Kimiaei
Fakultät für Mathematik, Universität Wien, Oskar-Morgenstern-Platz 1, A-1090 Wien, Austria
WWW:http://www.mat.univie.ac.at/~kimiaei/
E-mail: kimiaeim83@univie.ac.at

A. Neumaier
Fakultät für Mathematik, Universität Wien, Oskar-Morgenstern-Platz 1, A-1090 Wien, Austria
WWW:http://www.mat.univie.ac.at/~neum/
E-mail: Arnold.Neumaier@univie.ac.at

P. Faramarzi
Department of Mathematics, Faculty of Science, Razi University, Kermanshah, Iran
E-mail: p.faramarzi2018@gmail.com

1 Introduction

In this paper, we discuss a new subspace technique for solving the unconstrained black box optimization problem

$$\begin{array}{l} \min\ f(x) \\ \text{s.t.}\ \ x \in \mathbb{R}^n, \end{array} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable with Lipschitz continuous gradients, but accessible only through function values contaminated by rounding errors. In particular, exact function values, gradients, Lipschitz constants, or other information about the structure of f are not available. However, a numerical gradient approximation can be estimated by a finite difference method. We denote by $\widetilde{g} := \widetilde{g}(x)$ the estimated gradient at x and by $\widetilde{f} := \widetilde{f}(x)$ the inexact function value available at x.

In general, the goal of local optimization is to find at least one point x that is an **ε-approximate stationary point** if the norm of the exact gradient $g(x)$ of f is $\leq \varepsilon$. This criterion cannot be verified under our assumptions. Instead, our algorithm tries to find a point whose exact function value (although unknown) is at least as good as the exact function value of an ε-approximate stationary point whose approximated function value has been evaluated, for a threshold $\varepsilon = O(\sqrt{\omega})$, where ω is a bound on the error in the function values. If f has only one stationary point and ω is sufficiently small, this guarantees that the point found is arbitrarily close to the minimizer (if there are only finitely many stationary points, a point with small gradient could be guaranteed by a variant with restarts).

Our algorithm uses line searches based on the inexact function values evaluated along the search directions based on quadratic models. This algorithm estimates the gradient of the objective function by a finite difference method and the Hessian matrix of the objective function by a non-traditional limited memory quasi-Newton method. We construct quadratic models in a subspace of the previous search directions and then solve these models approximately such that, if possible, reductions of the model function value and gradient norm can be enforced. If this is not possible, we solve these quadratic models in a full subspace. Assuming that the exact gradient norm is not small, we prove that the (unknown) exact gradient and the function value found by the line search satisfy the Wolfe conditions. Therefore, our algorithm finds (at an unknown time) a point whose unknown gradient norm is below a computable threshold related to the level of rounding errors. The point with the best inexact function value $\widetilde{f}(x)$ among the evaluated values then has the required properties.

We say that a solver is robust if it has the highest number of solved problems, and efficient if it has the lowest relative cost of function evaluations. We say a solver is competitive if it is both robust and efficient. In this paper, we design an algorithm to be competitive in low, medium and high dimensions compared to other well-known black box optimization solvers.

We improve the state of the art in several respects:

• We ensure that our search directions satisfy the approximate angle condition

$$\frac{\widetilde{g}(x)^T p}{\|\widetilde{g}(x)\|\|p\|} \leq -\Delta_{\mathrm{a}} < 0, \tag{2}$$

i.e., the angle between the direction p and the approximate gradient $\widetilde{g}(x)$ at the point x obtained from a finite difference method remains bounded away from 90°. Here $0 < \Delta_{\mathrm{a}} < 1$ is a given threshold for the approximate angle condition (2).

• We show numerically that the new subspace method is more efficient than the traditional limited memory BFGS quasi-Newton method in low and high dimensions, and that it is more efficient than the best solvers from the literature.

1.1 Related work

There are many different algorithms for solving the problem (1). They can be classified into **deterministic** and **randomized**. An excellent reference for the deterministic methods is Conn et al. [6]. More recently, Larson et al. [22] have given a comprehensive discussion of both deterministic and stochastic methods. A comprehensive treatment of black box optimization solvers can be found in [18,20,31]. As shown in [18,20], **UOBYQA** by Powell [30] and **BCDFO** by Gratton et al. [14] were the two competitive model-based solvers for low-dimensional problems ($n \leq 20$) and two Nelder–Mead algorithms, **NELDER** by Kelley [17] and **NMSMAX** by Higham [16], were also competitive for very small n. A more recent version of the **BFO** solver was described by Porcelli & Toint [29], which showed avrage behaviour in the noiseless case in low to high dimensions, but was robust only for large noise. Two recent randomized line search solvers are **VRBBO** by Kimiaei & Neumaier [20] and **VRBBON** by Kimiaei [18]. In [18,20], it was shown that **VRBBO** and **VRBBON** are robust and moderately efficient compared to other competing local and global solvers for low-dimensional problems, but are competitive for problems in medium and high dimensions for both noiseless and noisy cases. Recently, researchers have shown increased interest in applying quasi-Newton methods that use the finite difference method to estimate the gradient to solve unconstrained black box optimization problems, e.g. [3,4,10,20,23–25,27]. Although this method is very useful for small scale problems, full inverse matrices require a lot of memory. A common suggestion is to use a quasi-Newton method with limited memory instead of such a standard method, which has a low memory requirement, see Liu & Nocedal [24]. **FMINUNC** from the Matlab optimization toolbox uses Wolfe line search along standard quasi-Newton directions. This solver was efficient and moderately robust for low and medium dimensional problems (cf. [20, Section 7]) in the noiseless case, but not in the presence of high noise (cf. [18, Section 7]), since a finite difference method leads to misleading information and gives a bad quasi-Newton direction.

To estimate the gradient, Berahas et al. [2, Table 2] discussed three bounds on the number of function evaluations, step sizes, and the exact gradient norm for all existing randomized and deterministic methods, such that the reduction in the error of the gradient norm is asymptotically bounded by the reduction in the exact gradient norm.

Line search methods proceed iteratively by generating a sequence of estimated or exact step sizes in the hope of finding a good decrease in the function value. Their global convergence was proved by Al-Baali & Fletcher [1], Fletcher [10], and Nocedal & Wright [28]. Berahas et al. [2] discuss complexity bounds for nonconvex, convex, and strongly convex cases for the accuracy of two different gradient estimates. Other useful references for first and second order complexity bounds are Curtis et al. [7,8] and Gratton et al. [13] for gradient-based trust region algorithms and Grapiglia et al. [12] for gradient-based nonlinear step size control algorithms. Cartis et al. [5] introduced a non-monotone gradient-related algorithm for nonconvex smooth unconstrained optimization problems that finds at most $\mathcal{O}(\varepsilon^{-2})$ function and gradient evaluations where the 2-norm gradient of a point x is below ε in the worst case.

Berahas et al. [3] estimated noise in the same way as Hamming [15] did. This estimate was added to the Armijo line search method and used to estimate the gradient by the finite difference technique. However, the effects of noise in BFGS updating have not been studied. Recently, Xie et al. [32] showed that the Wolfe conditions can be satisfied with the exact function value and gradient when these conditions are satisfied with the inexact function and gradient and the exact gradient is not small. They modified the BFGS method with the Wolfe line search without estimating the noise and showed that this method converges to a neighborhood of the solution, while the effects of the noise are taken into account in BFGS updating.

Kimiaei et al. [21] have proposed a limited memory method with minimal memory requirements for bound constrained optimization using the exact gradient. In this paper, we propose an extension of such a limited memory method for unconstrained black box optimization. Our algorithm uses Wolfe line searches based on the inexact function values evaluated along the search directions based on quadratic models in the subspace of the previous search directions. These line searches are called **approximate Wolfe line searches**, where in each iteration, instead of estimating the gradient of the objective function, the **directional derivative** is estimated by a **new variant of finite difference methods only with one function evaluation**, and at termination, the gradient is estimated by a finite difference method with n function evaluations. Hence these approximate line searches are not costly for problems in medium and high dimensions. The Hessian matrix of the objective function is approximated by a non-traditional limited memory quasi-Newton method. We improve these line searches in a heuristic way when they fail in the presence of rounding errors. **Hoping to find reductions of the model function value and gradient norm**, we construct quadratic models in a subspace of the previous search directions and then solve these models approximately. When these reductions are not possible, we solve these quadratic models in a full subspace, resulting in non-traditional limited memory quasi-Newton directions. In order to obtain the complexity bound, it is important that we can estimate the extent of the reduction of the function value, which can be achieved by a variant of the Wolfe line search method [1] with inexact function values and approximated gradients, as discussed later. Although our algorithm stops when the estimated gradient norm is below a computable threshold related to the level of rounding errors, it finds a point whose exact gradient norm is below a computable threshold related to the level of rounding errors. In this case, the approximate Wolfe conditions are numerically satisfied for both the inexact function and the approximate gradient. Assuming that the discretization and function evaluation errors occur in the estimation of the gradient and that the exact gradient norm is not small, we show in the same way as Xie et al. [32] that the Wolfe conditions are satisfied with the exact function value and gradient, but with the difference that these conditions are **independent of the choice of search directions**. In fact, we take into account the effects of noise in an estimated Hessian update. In contrast to Xie et al. [32], however, we enforce that our non-traditional limited memory quasi-Newton direction enforces the approximate angle condition (2). Therefore, we need not worry about whether the approximate Wolfe conditions can produce a quasi-Newton direction with both the inexact function and the gradient whose angle is bounded away from 90° by the approximate gradient. Ignoring some parts of the Hessian information reduces the efficiency of the traditional limited memory technique. We attempt to show **numerically** that the **improved subspace method** is **more efficient and robust** than the **standard BFGS quasi-Newton method**, even for problems in low and medium dimensions, and than the traditional limited memory BFGS quasi-Newton method for low and high dimensions.

1.2 The Wolfe line search in exact arithmetic

Our new algorithm uses an inexact version of the Wolfe line search method of Al-Baali & Fletcher [1], which is proposed there assuming exact arithmetic; we therefore call it **WLS-exact**. This method finds a step size that not only leads to a sufficient reduction in the function value, but is also close to a minimizer of the objective function with respect to the step size. Each iteration of **WLS-exact** is the scheme (S2) in [1, Section 2].

Al-Baali & Fletcher [1] assume that the objective function is bounded below, i.e., $f(\alpha) \geq \overline{f}$ for $\alpha \geq 0$ and a given finite threshold $\overline{f}$. When function values and gradients are known exactly, **WLS-exact** enforces the **Wolfe conditions**

$$f(x + \alpha p) \leq f(x) + \rho \alpha g(x)^T p, \tag{3}$$

$$|g(x+\alpha p)^T p| \le -\sigma g(x)^T p, \tag{4}$$

where $\sigma > \rho > 0$. Let $\phi(\alpha) := f(x+\alpha p)$. Then the ρ-line intersects the line $f = \overline{f}$ at point

$$\mu := \frac{\overline{f} - f(x)}{\rho g(x)^T p} = \frac{\overline{f} - \phi(0)}{\rho \phi'(0)^T p}. \tag{5}$$

WLS-exact restricts the step sizes to $(0, \mu]$. It finds an interval with suitable step sizes that satisfy the sufficient descent condition (3), and then finds a desired step size within that interval that satisfies the curvature condition (4). Interpolation and extrapolation are used to generate a new step size. Let a_j (for $j = 1, 2, \cdots, \infty$) be a sequence of the current best points of the function ϕ satisfying the condition (3) but not the condition (4), let b_j (for $j = 1, 2, \cdots, \infty$) be a sequence of points violating either (3) or $\phi(b_j) \ge \phi(a_j)$ or both, and let α_j (for $j = 1, 2, \cdots, \infty$) be a sequence of the current trial points of the function ϕ. Then the interval (a_j, b_j) found by **WLS-exact** contains the acceptable points or the points that $f(\alpha_j) \le \overline{f}$. For $j = 1$, $a_1 = 0$, $b_1 = \infty$, and $0 < \alpha_1 \le \mu$. Let $0 < \tau_1 \le \tau_2 \le \frac{1}{2}$, $1 < \tau_3 \le \tau_4$, $0 < \tau_5 \le \tau_6 \le \frac{1}{2}$, and $a_j, b_j > 0$ for $j \ge 1$. Whenever the condition $\phi(\alpha_j) > \phi(0) + \alpha_j \rho \phi'(0)$ or $\phi(\alpha_j) \ge \phi(a_j)$ is satisfied, the new step size

$$\alpha_{j+1} \in T(a_j, \alpha_j) := \begin{cases} [a_j + \tau_1(\alpha_j - a_j), \alpha_j - \tau_2(\alpha_j - a_j)] & \text{if } a_j < \alpha_j, \\ [\alpha_j + \tau_2(a_j - \alpha_j), a_j - \tau_1(a_j - \alpha_j)] & \text{if } \alpha_j < a_j \end{cases} \tag{6}$$

is obtained and the interval is updated as $a_{j+1} = a_j$ and $b_{j+1} = \alpha_j$. If the condition (4) is not satisfied the new step size

$$\alpha_{j+1} \in E(a_j, \alpha_j, b_j) := \begin{cases} [\min(\tau_3\alpha_j, \mu), \min(\tau_4\alpha_j, \mu)] & \text{if } a_j < \alpha_j < b_j = \infty, \\ [\alpha_j + \tau_5(b_j - \alpha_j), b_j - \tau_6(b_j - \alpha_j)] & \text{if } a_j < \alpha_j < b_j \le \mu, \\ [b_j + \tau_6(\alpha_j - b_j), \alpha_j - \tau_5(\alpha_j - b_j)] & \text{if } b_j < \alpha_j < a_j \le \mu \end{cases} \tag{7}$$

is computed and the interval is updated as $a_{j+1} = \alpha_j$ and $b_{j+1} = b_j$. In (6) and (7), α_{j+1} can be found within the defined intervals by minimizing quadratic/cubic polynomial interpolations; for more details see [1, Section 3].

Proposition 1 *[1, Theorem 2.2]*
Given $\sigma \ge \rho > 0$ and the finite threshold value $\overline{f}$, one of the following statements is valid:
(i) **WLS-exact** *ends with $f(x + \alpha_j p) < \overline{f}$ for some $\alpha_j \in (0, \mu]$. Here μ comes from (5).*
(ii) **WLS-exact** *ends with α_j satisfying (3) and (4).*
(iii) **WLS-exact** *fails, each interval $I_j := (a_j, b_j)$, $j = 1, 2, \cdots, \infty$, contains an interval of acceptable points and there exists a limit point $\widehat{\alpha} \in I_j$, for all j, such that, for sufficiently large j, a_j are monotonically (not strictly) increasing, $\lim_{j\to\infty} a_j = \widehat{\alpha}$ and b_j are monotonically (not strictly) decreasing, $\lim_{j\to\infty} b_j = \widehat{\alpha}$, and $\widehat{\alpha}$ is an acceptable point as in (ii).*
In particular, if $\sigma > \rho > 0$, (iii) cannot occur and **WLS-exact** *ends. Since we do not know in I_j whether $a_j < b_j$ or not, we replace a_j by b_j whenever $b_j < a_j$ for some $j = 1, 2, \cdots, \infty$.*

1.3 Overview of the new method

This paper constructs a new deterministic solver for unconstrained black box optimization problems, called **SSBBO** for "subspace black box optimization". The basic version of **SSBBO** is a quasi-Newton

algorithm using a finite difference technique for the approximation of the gradient, the limited memory direction by KIMIAEI et al. [21], and the Wolfe line search by AL-BAALI & FLETCHER [1] for finding step sizes but with the inexact function values and the estimated gradients. We call this line search method the **approximate Wolfe line search** method. This method is not costly even for problems in high dimensions, since one function evaluation is required in each iteration to estimate the directional derivative by a **new variant of finite difference methods** and n function evaluations are required to estimate the gradient by the forward finite difference method only in the last iteration.

Assuming that discretization and rounding errors occur when the gradient is estimated by a finite difference method and the exact gradient norm is not small, we prove that the Wolfe line search conditions can be satisfied with both the exact gradient and the function value, while these conditions are numerically satisfied with both the inexact function and the gradient. In this case, we prove the complexity result for both the approximate Wolfe line search algorithm and the basic version of **SSBBO** for the general case regardless of the choice of search directions. The order of our complexity bound is consistent with that of BERAHAS et al. [2].

The enhancements that make **SSBBO** competitive are:
• The new subspace direction is computed by finding the solution to a linear problem in the subspace and then solving a simple optimization problem to find, if possible, **a decrease in both the model function value and its gradient norm**.
• In the presence of rounding errors, the subspace direction computed need not satisfy the approximate angle condition (2) and must be modified. We find a robust modification described in Subsection 3.1.
• The Wolfe line search algorithm [1] may fail in finite precision arithmetic. In this case, the step size is generated by a **heuristic formula**. Although the algorithm moves somewhat away from a minimizer in this case, it may approach a minimizer in the next few attempts. At the very least, this idea helps us avoid failure.

SSBBO is available at `https://www.mat.univie.ac.at/~neum/software/SSBBO`

In Section 2, we introduce a basic version of our algorithm, called **SSBBO-basic**, which uses the approximated Wolfe line search algorithm and then obtain the complexity results of **SSBBO-basic**.

Section 3 discusses new enhancements. In Subsection 3.1 the subspace information and its update are described. Also, the subspace directions are computed while trying to reduce the function value and gradient norm of the quadratic model. In Subsection 3.2 an improved version of the Wolfe line search algorithm [1] is described, but with the inexact function values and estimated gradients, and in Subsection 3.3 an improved version of **SSBBO-basic** is introduced.

Section 4 contains extensive numerical results. They show that **SSBBO** of the present authors is effective and can compete with the best solvers from the literature when all 568 unconstrained problems from the `CUTEst` collection of GOULD et al. [11] are tested, with dimensions ranging from 1 to 9000. Our conclusions are summarized in Section 5.

2 Complexity of a basic version of SSBBO

For the theoretical analysis we first describe a simplified algorithm **SSBBO-basic**, an algorithm for unconstrained black box optimization problems that uses line searches along directions whose angle with an approximate gradient estimated by the forward finite differences is bounded away from 90°. The line search algorithm, called **AWLS-basic**, is an adaptation of the Wolfe line search algorithm described in Section 1.2 to account for inexact functions values and gradients. Given $\widetilde{\sigma} > \widetilde{\rho} > 0$, this

algorithm enforces the **approximate Wolfe conditions**

$$\widetilde{f}(x+\alpha p) \leq \widetilde{f}(x) + \widetilde{\rho}\alpha\widetilde{g}(x)^T p, \tag{8}$$

$$|\widetilde{g}(x+\alpha p)^T p| \leq -\widetilde{\sigma}\widetilde{g}(x)^T p, \tag{9}$$

with the inexact function and the estimated gradient. The **sufficient descent** (8) guarantees $\widetilde{f}(x+\alpha p) < \widetilde{f}(x)$ while the **curvature condition** (9) guarantees that α is not too far from a minimizer of the function f. **AWLS-basic** is like **WLS-exact**, but with the approximate Wolfe conditions (8) and (9). In fact the gradient is estimated by the forward finite difference method, denoted by **FFD**.

FFD takes the current point x and its inexact function value $f(x)$ as input. It uses the tuning parameter $0 < \varepsilon_h < 1$ for adjusting the finite difference step size and returns the estimated gradient $g(x)$ at x.

Algorithm 1 FFD, **estimation of the gradient by the forward finite difference method**

1: **for** $i = 1, 2, 3, \cdots$ **do**

Compute the step size

2: Compute $h_i := \sqrt{\varepsilon_h}\,\text{sign}\,(x_i)\max\{|x_i|, 1\}$.

Approximation of the ith component of the gradient

3: Compute $\widetilde{g}_i := (\widetilde{g}(x))_i := \dfrac{\widetilde{f}(x+h_i e_i) - \widetilde{f}(x)}{h_i}$. ▷ e_i is the ith coordinate direction

4: **end for**

To simplify our notation in the next algorithm, we define $\widetilde{\phi}(\alpha) = \widetilde{f}(x+\alpha p)$ and $\widetilde{\phi}'(\alpha) = \widetilde{g}(x+\alpha p)^T p$. We know that estimating the gradient $\widetilde{g}(x+\alpha p)$ in (9) by **FFD** requires n additional function evaluations, which can be costly in high dimensions. To avoid this drawback, we compute $\widetilde{\phi}'(\alpha_j)$ directly using a new variant of the forward finite difference method with one function evaluation, called **FFDD**, except in the last iteration. Therefore, **AWLS-basic** produces step sizes that are not too far from a minimizer and speeds up the process of reaching a minimizer.

FFDD takes the search direction p, the trial point $x + \alpha p$, and the current inexact function value $\widetilde{\phi}(\alpha)$ as input and uses the tuning parameter $0 < \varepsilon_h < 1$ for adjusting the step size. It returns the directional derivative $\widetilde{\phi}'(\alpha)$ as output.

Algorithm 2 FFDD, **estimation of the directional derivative by a new variant of FFD**

Compute the step size

1: Compute $h := \varepsilon_h \dfrac{\|x+\alpha p\|}{\|p\|}$.

Approximation of the directional derivative $\widetilde{\phi}'(\alpha)$ directly

2: Compute $\widetilde{\phi}'(\alpha) = \widetilde{g}(x+\alpha p)^T p = \dfrac{\widetilde{\phi}(\alpha+h) - \widetilde{\phi}(\alpha)}{h}$.

AWLS-basic takes the current point x as input and returns a better point $x+\alpha p$, its inexact function value $\widetilde{\phi}(\alpha) = \widetilde{f}(x+\alpha p)$, and its estimated gradient $\widetilde{g}(x+\alpha p)$ as output. It uses the tuning parameters

$\widetilde{\sigma} > \widetilde{\rho} > 0$ and the given finite threshold $\overline{f}$.

Algorithm 3 AWLS-basic, **basic approximate Wolfe conditions**

Compute $\widetilde{\mu} = (\overline{f} - \widetilde{\phi}(0))/(\widetilde{\rho}\widetilde{\phi}'(0))$.
Initialize the initial interval $(a_0, b_0) = (0, \widetilde{\mu})$.
Choose $\alpha_0 \in (0, \infty)$.
for $j = 1, 2, 3, \cdots$ **do**
    Trying to enforce the sufficient descent condition (8)
    Compute $\widetilde{\phi}(\alpha_j)$.
    First stopping test
    if $\widetilde{\phi}(\alpha_j) \leq \overline{f}$ **then** ▷ desired decrease in $\widetilde{\phi}$ has been found
        Compute the gradient $\widetilde{g}(x + \alpha_j p)$ by **FFD** and terminate **AWLS-basic**.
    end if
    if $\widetilde{\phi}(\alpha_j) > \widetilde{\phi}(0) + \alpha_j \widetilde{\rho}\widetilde{\phi}'(0)$ or $\widetilde{\phi}(\alpha_j) \geq \widetilde{\phi}(a_j)$ **then**
        Choose $\alpha_{j+1} \in T(a_j, \alpha_j)$.
        Update $a_{j+1} = a_j$ and $b_{j+1} = \alpha_j$. ▷ update the interval
    else
        Approximate the directional derivative $\widetilde{\phi}'(\alpha_j)$ by **FFDD**
        Approximate $\widetilde{\phi}'(\alpha_j)$ by **FFDD**.
        Second stopping test: Is the curvature condition (9) satisfied?
        if $|\widetilde{\phi}'(\alpha_j)| \leq -\widetilde{\sigma}\widetilde{\phi}'(0)$ **then**
            Compute the gradient $\widetilde{g}(x + \alpha_j p)$ by **FFD** and terminate **AWLS-basic**.
        end if
        Extrapolation or interpolation to get the new step size?
        Set $a_{j+1} = \alpha_j$.
        if $(b_j - a_j)\widetilde{\phi}'(\alpha_j) < 0$ **then**
            Choose $\alpha_{j+1} \in E(a_j, \alpha_j, b_j)$ and set $b_{j+1} = b_j$. ▷ extrapolation has been done
        else
            Choose $\alpha_{j+1} \in T(a_j, \alpha_j)$ and set $b_{j+1} = a_j$. ▷ interpolation has been done
        end if
    end if
end for

SSBBO-basic takes the initial point x_0 as input and returns an optimum point $x_{\rm best}$ and its inexact function value $\widetilde{f}_{\rm best}$ as output. It uses the following tuning parameters:
$\Delta_{\rm a} \in (0,1)$ (tiny parameter for the approximate angle condition (2)),
$\widetilde{\sigma} > \widetilde{\rho} > 0$ (parameters for line search),
$0 < \varepsilon_h < 1$ (parameter for adjusting the finite difference step size h),
$0 < \varepsilon_g < 1$ (minimum threshold for the stopping test).

Algorithm 4 SSBBO-basic, **basic subspace method for black box optimization**

Initialization

Compute the initial inexact function value $\widetilde{f}_0 := \widetilde{f}(x_0)$.
Compute $\overline{f} = \widetilde{f}_0 - 10^8(1 + |\widetilde{f}_0|)$. ▷ this choice has been suggested by **FMINUNC**
Estimate the initial gradient $\widetilde{g}_0 = \widetilde{g}(x_0)$ by **FFD**.
for $\ell = 1, 2, 3, \cdots$ **do**

Stopping test

  if $\|\widetilde{g}_\ell\| \le \varepsilon_g$, **then** set $x_{\rm best} := x_\ell$, $\widetilde{f}_{\rm best} := \widetilde{f}_\ell$, and stop; **end if**

Computing the search direction

  Choose p_ℓ such that the approximate angle condition (2) holds.

Use **AWLS-basic** to find α_ℓ satisfying (8) and (9)

  Perform **AWLS-basic** for computing $x_{\ell+1} = x_\ell + \alpha_\ell p_\ell$, $\widetilde{f}_{\ell+1} = \widetilde{f}(x_{\ell+1})$, and $\widetilde{g}_{\ell+1} = \widetilde{g}(x_{\ell+1})$.
end for

We prove complexity bounds for **AWLS-basic** and **SSBBO-basic** under the following assumptions, using the Euclidean norm:
(A1) The level set $L(x_0) := \{x \in \mathbb{R}^n \mid f(x) \le f(x_0),\ x_0 \in \mathbb{R}^n\}$ of f at the starting point x_0 is compact.
(A2) There is a convex neighbourhood Ω of $L(x_0)$ such that in Ω, the objective function f is continuously differentiable and its gradient g is Lipschitz continuous, i.e., there exists a constant $L > 0$ such that

$$\|g(x) - g(y)\| \le L\ \|x - y\|, \quad \text{for all}\ \ x, y \in \Omega. \tag{10}$$

(A3) The uncertainty of the function value $\widetilde{f}(x)$ obtained by a noisy oracle is globally bounded by a small threshold $\omega > 0$

$$|\widetilde{f}(x) - f(x)| \le \omega, \quad \text{for all } x \in \mathbb{R}^n. \tag{11}$$

It is well-known that (A1) and (A2) imply that some global minimizer $\widehat{x}$ exists, and

$$\widehat{f} := \inf_{\ell \ge 0} f(x_\ell) \tag{12}$$

is attained there, hence finite. (A2) implies

$$\|f(x+s) - f(x) - g(x)^T s\| \le \frac{L}{2}\|s\|^2, \quad \text{for } x, x+s \in \Omega. \tag{13}$$

Theorem 1 *Assume that (A1)–(A3) hold and let p be the search direction. If $\varepsilon_h := \Theta(\omega)$ and*

$$c_1\sqrt{\omega} \le h\|p\| \le c_2\sqrt{\omega} \tag{14}$$

then the directional derivative approximation defined by

$$\widetilde{g}_p(x) := h^{-1}(\widetilde{f}(x+hp) - \widetilde{f}(x))$$

satisfies

$$|\widetilde{g}_p(x) - g(x)^T p| = \mathcal{O}(\sqrt{\omega}\|p\|). \tag{15}$$

In particular, the approximate gradient estimated by **FFD** *satisfies* (15).

Proof By (11) and (14),

$$h|\widetilde{g}_p(x) - g(x)^T p| = |\widetilde{f}(x+hp) - \widetilde{f}(x) - hg(x)^T p| \le 2\omega + |f(x+hp) - f(x) - hg(x)^T p| \le 2\omega + \frac{L}{2}h^2\|p\|^2,$$

hence by (14)

$$|\widetilde{g}_p(x) - g(x)^T p| \le \frac{2\omega}{h} + \frac{L}{2}h\|p\|^2 = O(\sqrt{\omega}\|p\|).$$

By choosing $p = e_i$ (coordinate direction) for $i = 1, \cdots, n$, we have $|\widetilde{g}_p(x) - g_i(x)| = \mathcal{O}(\sqrt{\omega})$ for $i = 1, \cdots, n$, resulting in $\|\widetilde{g}(x) - g(x)\| = \mathcal{O}(\sqrt{\omega})$. □

This result is a variant of results discussed in Berahas et al. [2, Section 2.1] for forward finite difference methods for estimating the gradient.

It is well known that optimization methods can get stuck in almost flat regions, and that the gradient for a local minimizer in finite precision arithmetic does not vanish. For a given threshold $\varepsilon > 0$, let $N(\varepsilon)$ be the number of function evaluations to find an ε-approximate stationary point, i.e., approximate with $\|g(x)\| \le \varepsilon$. Then, under the assumptions (A1)–(A3), **SSBBO-basic** uses at most $N(\varepsilon)$ function evaluations to find a point $x_{\rm best}$ whose function value $f(x_{\rm best})$ is at most

$$\sup\{f(x) \mid x \in \mathbb{R}^n, \quad f(x) \le f(x_0), \quad \text{and} \quad \|g(x)\| \le \varepsilon\}.$$

In fact, $x_{\rm best}$ is the final best point found by **SSBBO-basic**. As in [20], $x_{\rm best}$ is unknown to the algorithm because the gradients and Lipschitz constants are unknown. **SSBBO-basic** finds $x_{\rm best}$ whose function value $f(x_{\rm best})$ is equal to or better than a point whose gradient was small. If the gradients are small only near a global optimizer, a point near the local optimizer is found. If some iterate passes close to a non-global local optimizer or a saddle point, **SSBBO-basic** may escape from its neighbourhood. In this case, only a variant with restarts would lead to convergence to a point with a small gradient.

We now discuss how to get the complexity result for **AWLS-basic**. It plays a key role in proving the complexity bound for **SSBBO-basic**.

We deduce from Theorem 1 that a complexity bound for **AWLS-basic** can be obtained .

Proposition 2 *Assume that (A1)–(A3) hold, the gradient of f is estimated by* **FFD**, *the direction p_ℓ satisfies the approximate angle condition* (2), *and the step size α satisfies the strong Wolfe conditions* (8) *and* (9). *Denote by $\widetilde{\theta}$ the angle between $\widetilde{g}(x)$ and p and by θ the angle between $g(x)$ and p. Then if $\varepsilon_h := \Theta(\omega)$ and h satisfies* (14)*:*
(i) For all $x,\ y \in \mathbb{R}^n$ we have

$$\|\widetilde{g}(x) - \widetilde{g}(y)\| = \mathcal{O}(\sqrt{\omega}) + L\|x - y\|.$$

(ii) p satisfies the angle condition for the exact gradient

$$\frac{g(x)^T p}{\|g(x)\|\|p\|} < -\frac{\Delta_{\mathrm{a}}}{2}. \tag{16}$$

(iii) Under the assumptions

$$\|\widetilde{g}(x)\| > \frac{2\sqrt{\omega}}{\Delta_{\mathrm{a}}(1-\sigma)}, \tag{17}$$

$$\|g(x)\| \geq \max\Big\{\frac{4\widetilde{\rho}\sqrt{\omega}}{\rho\Delta_{\mathrm{a}}}, \frac{(1+\widetilde{\sigma})\sqrt{\omega}}{\sigma\Delta_{\mathrm{a}}}\Big\}, \tag{18}$$

and

$$\|\widetilde{g}(x)\|\|g(x)\| \geq \frac{16L\omega}{\rho(1-\widetilde{\sigma})\Delta_{\mathrm{a}}^2}, \tag{19}$$

α satisfies the strong Wolfe conditions

$$f(x+\alpha p) \leq f(x) + \rho_1 \alpha g(x)^T p, \tag{20}$$

$$|g(x+\alpha p)^T p| \leq -\sigma_1 g(x)^T p, \tag{21}$$

where $0 < \rho_1 := \widetilde{\rho} - \rho < \widetilde{\rho} < \widetilde{\sigma} < \sigma_1 := \widetilde{\sigma} + \sigma$.
(iv) Given $0 < \rho_1 < \sigma_1$ *and the minimal threshold* $\alpha_{\min} \in (0,1)$ *for the step size, we define*

$$\overline{\mu} := \overline{\mu}(x) = \frac{\overline{f} - f(x)}{\rho_1 g(x)^T p}. \tag{22}$$

and choose the step size $\alpha \in (0, \overline{\mu}]$, *for* $\ell \geq 1$, *and* $\tau \in (0,1)$. *Then* **AWLS-basic** *needs at most*

$$K := K(x) = \Big\lceil \frac{\log \alpha_{\min}/\overline{\mu}(x)}{\log \tau} \Big\rceil \tag{23}$$

iterations and at most

$$N^f := N^f(x) = 2K(x) + n \tag{24}$$

function evaluations to satisfy (20) *and* (21) *(The factor n comes from the approximated gradient evaluations needed in the last iteration of Wolfe line search).*

Proof (i) From (A2) and Theorem 1, we have

$$\|\widetilde{g}(x) - \widetilde{g}(y)\| = \|\widetilde{g}(x) - g(x) + g(x) - g(y) + g(y) - \widetilde{g}(y)\| = \mathcal{O}(\sqrt{\omega}) + L\|x-y\|.$$

(ii) By [32, Lemma 3.11], since $\cos\widetilde{\theta} \geq \Delta_{\mathrm{a}} > 0$ by (2) and (15) holds, $\cos\theta \geq \Delta_{\mathrm{a}}/2$ and so (16) is obtained.
(iii) By (i), Cauchy–Schwarz inequality, and (9), we obtain

$$\begin{aligned}(\mathcal{O}(\sqrt{\omega}) + \alpha L\|p\|)\|p\| &= \|\widetilde{g}(x+\alpha p) - \widetilde{g}(x)\|\|p\| \\ &\geq (\widetilde{g}(x+\alpha p) - \widetilde{g}(x))^T p \geq (\widetilde{\sigma} - 1)\widetilde{g}(x)^T p,\end{aligned}$$

resulting in

$$\alpha \geq \frac{1-\widetilde{\sigma}}{L}\Big(\frac{-\widetilde{g}(x)^T p}{\|p\|^2}\Big) - \frac{\mathcal{O}(\sqrt{\omega})}{L\|p\|} \geq \frac{(1-\widetilde{\sigma})\Delta_{\mathrm{a}}\|\widetilde{g}(x)\| - \sqrt{\omega}}{L\|p\|} \geq \frac{(1-\widetilde{\sigma})\Delta_{\mathrm{a}}\|\widetilde{g}(x)\|}{2L\|p\|}.$$

(iii) Under the condition (18), the statements (ii) and (iii) result in

$$-\rho\alpha g(x)^T p \geq \frac{\rho(1-\widetilde{\sigma})\Delta_{\mathrm{a}}\|\widetilde{g}(x)\|}{2L\|p\|} g(x)^T p \geq \frac{\rho(1-\widetilde{\sigma})\Delta_{\mathrm{a}}\|\widetilde{g}(x)\|}{2L\|p\|}\|g(x)\|\|p\|\cos\theta$$
$$\geq \frac{\rho(1-\widetilde{\sigma})\Delta_{\mathrm{a}}^2}{4L}\|\widetilde{g}(x)\|\|g(x)\| \geq 4\omega. \tag{25}$$

On the other hand, (18) gives

$$-\rho\alpha g(x)^T p \geq \frac{\rho\Delta_{\mathrm{a}}}{2}\alpha\|p\|\|g(x)\| \geq \frac{\rho\Delta_{\mathrm{a}}}{2}\alpha\|p\|\Big(\frac{4\widetilde{\rho}\sqrt{\omega}}{\rho\Delta_{\mathrm{a}}}\Big) = 2\widetilde{\rho}\alpha\|p\|\sqrt{\omega}. \tag{26}$$

By summing both sides of (25) with (26), it results in

$$-\rho\alpha g(x)^T p \geq 2\omega + \widetilde{\rho}\alpha\|p\|\sqrt{\omega}.$$

From (8) and (A3), we get

$$f(x+\alpha p) \leq f(x) + 2\omega + \widetilde{\rho}\alpha\widetilde{g}(x)^T p \leq f(x) + 2\omega + \widetilde{\rho}\alpha\Big(g(x)^T p + \|p\|\sqrt{\omega}\Big)$$
$$= f(x) + 2\omega + \widetilde{\rho}\alpha\|p\|\sqrt{\omega} + \widetilde{\rho}\alpha g(x)^T p$$
$$\leq f(x) + (\widetilde{\rho}-\rho)\alpha g(x)^T p = f(x) + \rho_1\alpha g(x)^T p;$$

hence (20) holds. From (9) we conclude that

$$g(x+\alpha p)^T p + \|p\|\sqrt{\omega} \geq \widetilde{g}(x+\alpha p)^T p \geq \widetilde{\sigma}\widetilde{g}(x)^T p \geq \widetilde{\sigma}\Big(g(x)^T p - \|p\|\sqrt{\omega}\Big),$$

so that

$$g(x+\alpha p)^T p \geq \widetilde{\sigma} g(x)^T p - (1+\widetilde{\sigma})\|p\|\sqrt{\omega} \geq (\widetilde{\sigma}+\sigma) g(x)^T p = \sigma_1 g(x)^T p$$

by (18); hence (21) is obtained.
(iv) In the worst case, neither Theorem 1(i) nor Theorem 1(ii) is satisfied. In this case, Theorem 1(iii) holds such that the limit point $\widehat{\alpha}$ of $I_j := (a_j, b_j)$ enforces the Wolfe condition (3) and (4) for sufficiently large j. To obtain a bound on the iterations, we use the condition given by interpolation

$$|b_j - a_j| \leq (1-\tau_1)|b_{j-1} - a_{j-1}| \ \text{ with } \tau_1 \in (0,1) \tag{27}$$

and the condition given by the extrapolation

$$|b_j - a_j| \leq (1-\tau_2)|b_{j-1} - a_{j-1}| \ \text{ with } \tau_2 \in (0,1). \tag{28}$$

The condition (27) is [1, (16)] obtained from (6) and the condition (28) is [1, (17)] obtained from (7). Denoting $\tau := \min\{1-\tau_1, 1-\tau_2\} \in (0,1)$, we have

$$|b_j - a_j| \leq \tau|b_{j-1} - a_{j-1}|. \tag{29}$$

We assume that **AWLS-basic** terminates after K iterations. Applying the inequality (29) recursively, for sufficiently large j we get

$$\alpha_{\min} \leq \widehat{\alpha} = |b_K - a_K| \leq \tau^{K-1}|b_1 - a_1| \leq \tau^{K-1}|\overline{\mu} - a_1| \leq \tau^{K-1}\overline{\mu},$$

so that the number is bounded by (23) and hence the number of function evaluations is bounded by (24), since $2K+n$ function evaluations are required. One of the stopping tests (in Lines 6-8 and 14-16 of **AWLS-basic**) requires n function evaluations in the last iteration to estimate the gradient with **FFD**, and two function evaluations in Lines 5 and 13 to compute the function value and directional derivative with **FFDD** in each iteration. Note that since it is assumed that Theorem 1(i) does not hold, $b_1 \leq \overline{\mu} < \infty$. Otherwise, if $b_1 = \infty$, no bracket is found and extrapolation is performed until α_j reaches $\overline{\mu}$. Then $f(\overline{\mu}) \leq \overline{f}$ and **AWLS-basic** ends. □

This leads to the complexity bound for **SSBBO-basic**.

Theorem 2 *Suppose that assumptions (A1)–(A3) hold and let f_0 be the initial function value of f. Given $0 < \mu_{\min} < \infty$, define*

$$\overline{\mu}_{\min} := \min_{\ell \geq 0}(\mu_{\min}, \overline{\mu}_\ell) \quad \text{with } \overline{\mu}_\ell := \overline{\mu}(x_\ell) = \frac{\overline{f} - f(x_\ell)}{\rho_1 g(x_\ell)^T p_\ell}.$$

Here ρ_1 comes from Proposition 2(iii) and $\overline{f}$ comes from Line 2 of Algorithm 4. Then **SSBBO-basic** *needs at most $\mathcal{O}(\omega^{-1})$ iterations and $\mathcal{O}((2\overline{K} + n)\omega^{-1})$ function evaluations to find a point x with $\|g(x)\| = \mathcal{O}(\sqrt{\omega})$. Here*

$$K_\ell := K(x_\ell) \leq \overline{K} := \left\lceil \frac{\log \alpha_{\min}/\overline{\mu}_{\min}}{\log \tau} \right\rceil,$$

$\alpha_{\min}$ and τ come from Proposition 2, and the factor n comes from the approximate gradient evaluations needed in the last iteration of **AWLS-basic**.

Proof Let $\mathcal{S}$ be the set of iterations generated by **AWLS-basic** such that

$$\|g(x_\ell)\| > \max\left\{\frac{4\widetilde{\rho}}{\rho \Delta_{\mathrm{a}}}, \frac{(1+\widetilde{\sigma})}{\sigma \Delta_{\mathrm{a}}}\right\}\sqrt{\omega} = \mathcal{O}(\sqrt{\omega}) \quad \text{for all } \ell \in \mathcal{S};$$

$\mathcal{S}$ is not empty by Proposition 1. In other words, all iterations of **SSBBO-basic** decrease the function value; hence, they are successful. Based on Proposition 1, we consider the two following cases:
Case 1. Suppose that the statement (i) in Proposition 1 holds. Then, we get

$$f_{\ell+1} := f(x_\ell + \alpha_\ell p_\ell) \leq \overline{f}, \quad \text{for some } \alpha_\ell \in (0, \overline{\mu}_\ell];$$

hence

$$f_\ell - f_{\ell+1} \geq f_\ell - \overline{f} \geq 0. \tag{30}$$

Case 2. Suppose that the statement (ii) in Proposition 1 holds. Then, by the angle condition (2), we get from Proposition 2

$$f_\ell - f_{\ell+1} \geq -\rho_1 \alpha_\ell g_\ell^T p_\ell \geq \frac{4\rho_1 \omega}{\rho}. \tag{31}$$

Then (30) and (31) result in $f_\ell - f_{\ell+1} \geq \dfrac{4\rho_1\omega}{\rho}$. Summing both sides and using (12) gives

$$f_0 - \widehat{f} \geq \sum_{\ell \in \mathcal{S}}(f_\ell - f_{\ell+1}) \geq |\mathcal{S}|\frac{4\rho_1\omega}{\rho},$$

leading to $|\mathcal{S}| \leq \dfrac{\rho(f_0 - \widehat{f})}{4\rho_1\omega}$. Denote by $N_\ell^f = N^f(x_\ell)$ the number of function evaluations in each call to **AWLS-basic** for $\ell \geq 1$. By Proposition 2(iii), the total number of function evaluations by **SSBBO-basic** is

$$N(\omega) = \sum_{\ell \in \mathcal{S}} N_\ell^f = \sum_{\ell \in \mathcal{S}}(2K_\ell + n) \leq (2\overline{K} + n)|\mathcal{S}| = \mathcal{O}\Big((2\overline{K} + n)\omega^{-1}\Big).$$

□

The order of our bound is the same as that found by Berahas et al. [2].

3 Enhancements

In this section we discuss how new improvements make **SSBBO-basic** very competitive. A limited memory quasi-Newton direction and an update to relevant subspace information are given, leading to an algorithm for computing a subspace step that reduces, if possible, both the value of the model function and its gradient norm. The basic formulation of the subspace direction comes from [21]. We modify the raw subspace direction to enforce that the angle away from 90° is bounded by an approximate gradient estimated by forward finite differences. We discuss an improved Wolfe line search algorithm enriched with a heuristic formula that helps the algorithm to avoid failure. Finally, the **SSBBO** algorithm is explained.

3.1 Decreasing model function value and its gradient norm

In this subsection, we give some details about subspace information, the limited memory quasi-Newton direction, and how to update the subspace information. Then, using a subspace technique, we try to find decreases in the function value and gradient norm of quadratic models. This can be done by solving a linear problem and a one-dimensional bound constrained problem in a cheap way. Our subspace direction is computed by an algorithm called **subspaceDir**. The basic version of this algorithm has already been proposed in [21, Section 2.2], but with the exact gradient, while here the gradient is estimated by **FFD**. Indeed, the new ingredient of **subspaceDir** is to find reductions of the function value and the gradient norm of the quadratic models by new subspace steps.

Let S_ℓ be the $n \times m$ matrix whose columns are (in the actual implementation a permutation of) the previous m search directions,

$$S_\ell := \{s_{\ell-m+1}, \cdots, s_\ell\} = \{x_{\ell-m+1} - x_{\ell-m}, \cdots, x_\ell - x_{\ell-1}\}, \tag{32}$$

and $Y_\ell \in \mathbb{R}^{n\times m}$ be the corresponding gradient differences,

$$Y_\ell := \{y_{\ell-m+1}, \cdots, y_\ell\} = \{\widetilde{g}_{\ell-m+1} - \widetilde{g}_{\ell-m}, \cdots, \widetilde{g}_\ell - \widetilde{g}_{\ell-1}\}. \tag{33}$$

We first describe the raw subspace direction corresponding to the limited memory quasi-Newton direction described in [21, Section 2.2] but with the gradient estimated by **FFD**. We denote by `YY` the componentwise squares of Y_ℓ, by `SS` the componentwise squares of S_ℓ , by // the componentwise division, and denote $\overline{d} := \sum_{i=1}^m \texttt{YY}_{:i}$ and $\underline{d} := \sum_{i=1}^m \texttt{SS}_{:i}$. Moreover we define the matrices $U_\ell := Y_\ell - D_\ell S_\ell$, $D_\ell := \mathrm{diag}(d_\ell)$ with $d_\ell := \sqrt{\overline{d}//\underline{d}}$, and $\Sigma_\ell := U_\ell^T S_\ell$ (which is symmetric). As in [21], $B_\ell := D_\ell + U_\ell \Sigma_\ell^{-1} U_\ell^T$ is an approximate Hessian matrix, determined by unique S_ℓ and Y_ℓ and the **quasi-Newton condition**

$$B_\ell S_\ell = Y_\ell. \tag{34}$$

If we define the matrix $H_\ell = S_\ell^T Y_\ell$, we obtain from (34) a further approximation by

$$H_\ell = S_\ell^T Y_\ell = S_\ell^T B_\ell S_\ell. \tag{35}$$

We define the matrix $M_\ell := Y_\ell^T D_\ell^{-1} Y_\ell - H_\ell$ and obtain the solution of the linear system $M_\ell z_\ell = U_\ell^T D_\ell^{-1} \widetilde{g}_\ell$. Here $\widetilde{g}_\ell$ is estimated by **FFD**, but is exact in [21]. Then [21, Theorem 1] shows that p_ℓ computed by

$$p_\ell := D_\ell^{-1}(U_\ell z_\ell - \widetilde{g}_\ell), \tag{36}$$

is the solution of $B_\ell p_\ell = -\widetilde{g}_\ell$.

We then update the ingredients of the subspace by **updateSY**. **updateSY** takes the matrices $S_{\ell-1}$, $Y_{\ell-1}$, and $H_{\ell-1}$ and the subspace dimension m as input and uses the tuning parameter $m_{\max}$ as an upper bound for m. It returns the updated matrices S_ℓ, Y_ℓ, H_ℓ and the subspace dimension m.

Algorithm 5 updateSY, update the subspace

if $m < m_{\max}$ **then**
    Update $m = m + 1$.
else
    Set $m = 1$.
end if
Replace the mth column of $S_{\ell-1}$ by s_ℓ, resulting it S_ℓ.
Replace the mth column of $Y_{\ell-1}$ by y_ℓ, resulting in Y_ℓ.
Replace the mth row of H_ℓ by $y_\ell^T S_{\ell-1}$ and the mth column of H_ℓ by $S_{\ell-1}^T y_\ell$, resulting H_ℓ.

In practice, we don't compute B_ℓ, only S_ℓ, Y_ℓ, and H_ℓ are used to compute the subspace directions. **updateSY** enforces that H_ℓ becomes symmetric.

Let us describe what is new in **subspaceDir**. We show how to minimize the maximal norm of the gradient model in the subspace spanned by the columns of the matrix $S_\ell \in \mathbb{R}^{n\times m}$, where typically $m \ll n$. We define with $\widetilde{f}_\ell := \widetilde{f}(x_\ell)$ the inexact function value at the current point x_ℓ, with $\widetilde{g}_\ell := \widetilde{g}(x_\ell)$ the estimated gradient function at x_ℓ, and with

$$c_\ell := S_\ell^T \widetilde{g}_\ell \tag{37}$$

the estimated gradient function restricted to S_ℓ at x_ℓ. Then the quadratic model

$$\widetilde{f}(x_\ell + S_\ell z) - \widetilde{f}_\ell \approx q(z) := c_\ell^T z + \frac{1}{2} z^T H_\ell z$$

is constructed whose gradient is

$$q'(z) := \partial q(z)/\partial z := c_\ell + H_\ell z.$$

Here H_ℓ has already been defined by (35) and c_ℓ by (37). The stationary point is at

$$\widehat{z} := -H_\ell^{-1} c_\ell. \tag{38}$$

To find a point that reduces not only the model function value q but also, if possible, its gradient norm $\|q\|$, we perform an exact line search on the model function along $\widehat{z}$ by solving the simpler univariate problem

$$\begin{aligned} &\min_\beta \ \|q'(\beta\widehat{z})\|_2 = |1-\beta|\|c_\ell\|_2 \\ &\text{s.t.} \ \ q(\beta) = \gamma_1\beta + \gamma_2\beta^2 \le 0, \end{aligned} \tag{39}$$

where

$$\gamma_1 := c_\ell^T \widehat{z}, \quad \gamma_2 := \frac{1}{2}\widehat{z}^T H_\ell \widehat{z}. \tag{40}$$

In three situations, we do not want to improve the model gradient norm:
• If γ_1 or γ_2 is contaminated by NaN or $\pm\infty$, the constraint is not satisfied.
• If $\gamma_2 \le 0$, $q(\beta)$ is flat or unbounded below.

• If $\gamma_2 > 0$ but $\gamma_1 \geq 0$, the minimal q is attained at $\beta = 0$.
In these cases, the problem (39) cannot be solved and p_ℓ is computed by (36). In the remaining case, $\gamma_1 < 0 < \gamma_2$ and $\beta^* := -\gamma_1/(2\gamma_2) > 0$. Since $q(\beta) \leq 0$ iff $0 \leq \beta \leq 2\beta^*$, the constraint in (39) is equivalent to $0 \leq \beta \leq 2\beta^*$. This case is acceptable only if β^* is a real value (otherwise p_ℓ is computed by (36)). Consequently, (39) reduces to

$$\begin{aligned} &\min\ |1-\beta| \\ &\text{s.t.}\ \ 0 \leq \beta \leq 2\beta^* \end{aligned}$$

with the solution $\widehat{\beta} := \min(1, 2\beta^*)$. We denote by $\widehat{q} := q(\widehat{\beta})$ the optimal model function value. Indeed, $\widehat{\beta}$ does not necessarily reduce both the function value q and the gradient norm q' of the model in finite precision arithmetic. To check this, we define the computational measure $\mathtt{df}_\ell$, which is initialized by $\mathtt{df}_0 := \varepsilon_1 |\widetilde{f}_0|$ with $\varepsilon_1 > 0$ (if $\widetilde{f}_0 = 0$ set $\mathtt{df}_0 := 1$) and updated based on the information of the function value

$$\mathtt{df}_\ell := \begin{cases} \gamma_f^1(\widetilde{f}_\ell - \widetilde{f}_{\ell-1}) & \text{if } \widetilde{f}_\ell < \widetilde{f}_{\ell-1} - \mathtt{df}_{\ell-1}, \\ \max\{\gamma_f^2 \mathtt{df}_{\ell-1}, \gamma_f^3(|\widetilde{f}_\ell| + |\widetilde{f}_{\ell-1}|)\} & \text{otherwise} \end{cases}$$

Here, the parameters $\gamma_f^1 \in (0,1)$, $\gamma_f^2 > 1$, and $\gamma_f^3 \in (0,1)$ are tuning parameters. At the ℓth iteration, when $\widehat{q} \leq -\mathtt{df}_\ell$ we have a good decrease in both the model function value and its gradient norm in the current subspace. Then we can compute the **new subspace direction** by

$$p_\ell := \widehat{\beta} S_\ell \widehat{z}. \tag{41}$$

To be efficient and robust, **AWLS-basic** must bound the angle between the approximate gradient computed by **FFD** and the search direction computed by **subspaceDir** away from 90°. Inspired by [21, Proposition 1] but using an approximate gradient, we modify the search direction $\overline{p}_\ell = p_\ell$, which can be obtained from either (36) or (41), if this direction does not satisfy the approximate angle condition (2). In this case, **subspaceDir** computes $\eta_1 = \widetilde{g}_\ell^T \widetilde{g}_\ell$, $\eta_2 = \overline{p}_\ell^T \overline{p}_\ell$, and $\eta = \widetilde{g}_\ell^T \overline{p}_\ell$. Then it finds t satisfying

$$\frac{\eta - t\eta_1}{\sqrt{\eta_1(\eta_2 - 2t\eta + t^2\eta_1)}} \leq -\Delta_{\mathrm{a}} \in (0,1), \tag{42}$$

and recomputes the search direction by $p_\ell = \overline{p}_\ell - t\widetilde{g}_\ell$ such that the approximate angle condition (2) holds; but sometimes it is not possible due to rounding errors. In this case, **subspaceDir** recomputes the search direction by a **diagonal preconditioned steepest descent direction** $p_\ell = -D_\ell^{-1}\widetilde{g}_\ell$.

In the ℓth iteration, **subspaceDir** takes the matrices S_ℓ, Y_ℓ, H_ℓ and the estimated gradient $\widetilde{g}_\ell$ as input and returns the subspace direction p_ℓ as output. It uses the tuning parameter $0 < \Delta_{\mathrm{a}} < 1$ for the approximate angle condition.

Algorithm 6 subspaceDir, a subspace direction

Obtain the solution of quadratic model in the subspace

Compute $\widehat{z} = -H_\ell^{-1} c_\ell$.

Minimize $|1-\beta|$ subject to $0 \le \beta \le 2\beta^*$ to obtain $\widehat{\beta}$

Compute $\gamma_1 = c_\ell^T \widehat{z}$, $\gamma_2 = \frac{1}{2}\widehat{z}^T H_\ell \widehat{z}$, $\beta^* = -\gamma_1/(2\gamma_2) > 0$ and $\widehat{\beta} = \min(1, 2\beta^*)$.

Is a reduction in the function value and gradient norm of the quadratic model?

if $\widehat{q} \le -\mathtt{df}_\ell$ **then** ▷ a reduction is found

  Compute $p_\ell = \widehat{\beta} S_\ell \widehat{z}$.

else

  Compute the diagonal matrix $D_\ell := \mathrm{diag}(d_\ell)$ with $d_\ell = \sqrt{\overline{d}/\underline{d}}$, $\overline{d} = \sum_{i=1}^m \mathtt{YY}_{:i}$, $\underline{d} = \sum_{i=1}^m \mathtt{SS}_{:i}$.

  Compute the matrix $U_\ell = Y_\ell - D_\ell S_\ell$.

  Compute the matrix $M_\ell = Y_\ell^T D_\ell^{-1} Y_\ell - H_\ell$.

  Solve the linear system $M_\ell z_\ell = U_\ell^T D_\ell^{-1} \widetilde{g}_\ell$.

  Compute the subspace direction by $p_\ell = D_\ell^{-1}(U_\ell z_\ell - \widetilde{g}_\ell)$.

end if

Enforcing the angle condition

if $\frac{\widetilde{g}_\ell^T p_\ell}{\|\widetilde{g}_\ell\| \|p_\ell\|} \le -\Delta_{\mathrm{a}} < 0$ **then** ▷ $\Delta_{\mathrm{a}} \in (0,1)$ is a tuning parameter

  Set $\eta_1 = \widetilde{g}_\ell^T \widetilde{g}_\ell$, $\eta_2 = p_\ell^T p_\ell$, and $\eta = \widetilde{g}_\ell^T p_\ell$. Then find t satisfying (42).

  if t is numerically computable **then** ▷ but sometimes this is not possible due to rounding errors

    Calculate $p_\ell = p_\ell - t\widetilde{g}_\ell$.

  else

    Calculate $p_\ell = -D_\ell^{-1} \widetilde{g}_\ell$. ▷ **diagonal preconditioned steepest descent direction**

  end if

end if

3.2 An improved Wolfe line search

The statements (i) and (ii) in Proposition 1 may not be numerically satisfied due to rounding errors. In this case, **AWLS-basic** must be improved. To prevent the failure of **AWLS-basic**, it is proposed to use the heuristic formulas proposed in [21, Section 3.1], but the gradient is estimated by **FFD**. Although the algorithm moves somewhat away from a minimizer in this case, it may approach a minimizer in the next few attempts. At the very least, this idea helps us avoid failure.

An improved version of **AWLS-basic** by Al-Baali & Fletcher [1] is called **AWLS**. It uses the following tuning parameters:
$\widetilde{\sigma} > \widetilde{\rho} > 0$ (parameters for line search),
$0 < \Delta_\alpha < 1$ (tiny parameter for adjusting the heuristic step size),
$0 < \varepsilon_h < 1$ (parameter for adjusting the finite difference step size h).

Algorithm 7 AWLS, an improved approximate Wolfe line search

Perform **AWLS-basic** in finite precision arithmetic

Call **AWLS-basic** to find a step size satisfying the approximate Wolfe line search conditions (8) and (9).

Find the step size by a heuristic way when **AWLS-basic** fails in finite precision arithmetic

if AWLS-basic does not enforce the approximate Wolfe conditions **then**
  Find $\texttt{ind} := \{i \mid p_i \neq 0\}$.
  if both x and p are not zero **then**
    Compute $\alpha := \Delta_\alpha |\widetilde{f}/\widetilde{g}^T p|$.
  else if only p is not zero **then**
    Compute $\alpha := \Delta_\alpha \max(|\widetilde{f}/\widetilde{g}^T p|, \min\{|x_i/p_i| \mid i \in \texttt{ind}\})$.
  else
    Choose $\alpha := 1$.
  end if
  Compute the new point $x + \alpha p$ and its inexact function value $\widetilde{f}(x + \alpha p)$.
  Estimate the gradient $\widetilde{g}(x + \alpha p)$.
end if

3.3 The new subspace algorithm

Our new algorithm is called **subspace method for unconstrained black box optimization** (**SSBBO**). This algorithm takes advantage of the new subspace technique, which attempts to significantly reduce not only the value of the model function, but also its gradient norm. As long as **SSBBO** cannot reach an ε-approximate stationary point, **subspaceDir** is used to compute the direction along which **AWLS** is tried. Then some necessary information about the subspace is updated.

SSBBO takes the initial point x_0 and maximal function evaluations (`nfmax`) as input and returns the best point $x_{\rm best}$ and its inexact function value $\widetilde{f}_{\rm best}$ as output. It uses the following tuning parameters:
$\varepsilon_1 > 0$ (parameter for adjusting the initial `df`),
$\gamma_f^2 > 1$ (parameter for expanding `df`),
$\gamma_f^1, \gamma_f^3 \in (0,1)$ (parameters for reducing `df`),
$m > 0$ (memory for the subspace),
$\Delta_{\rm a} \in (0,1)$ (tiny parameter for the approximate angle condition (2)),
$0 < \Delta_\alpha < 1$ (tiny parameter for adjusting the heuristic step size),
$\widetilde{\sigma} > \widetilde{\rho} > 0$ (parameters for line search),
$0 < \varepsilon_h < 1$ (parameter for adjusting the finite difference step size h).

Algorithm 8 SSBBO, subspace method for black box optimization

Initialization
Compute the initial inexact function value $\widetilde{f}_0$.
Compute $\overline{f} = \widetilde{f}_0 - 10^8(1 + |\widetilde{f}_0|)$. ▷ this choice has been suggested by **FMINUNC**
Estimate the gradient $\widetilde{g}_0$ by **FFD**.
Initialize $\texttt{df}_0 = \varepsilon_1 |\widetilde{f}_0|$.
Set $x_{\rm best} := x_0$ and $\widetilde{f}_{\rm best} := \widetilde{f}_0$.
for $\ell = 1, 2, 3, \cdots$ **do**
    Compute the subspace direction p_ℓ by **subspaceDir**.
    Find step size α_ℓ
    Perform **AWLS** to find α_ℓ resulting in $x_{\ell+1}$, $\widetilde{f}_{\ell+1}$, and $\widetilde{g}_{\ell+1}$.
    Stopping test
    if `nfmax` is reached **then**
        Set $x_{\rm best} := x_{\ell+1}$ and $\widetilde{f}_{\rm best} := \widetilde{f}_{\ell+1}$; **stop**.
    end if
    Update the predicated decrease $\texttt{df}_{\ell+1}$ in the function value
    if $\widetilde{f}_{\ell+1} < \widetilde{f}_\ell - \texttt{df}_\ell$ **then**
        $\texttt{df}_{\ell+1} = \gamma_f^1(\widetilde{f}_{\ell+1} - \widetilde{f}_\ell)$.
    else
        $\texttt{df}_{\ell+1} = \max\{\gamma_f^2 \texttt{df}_\ell, \gamma_f^3(|\widetilde{f}_{\ell+1}| + |\widetilde{f}_\ell|)$.
    end if
    Update the ingredients of the subspace
    Set $s_\ell := x_{\ell+1} - x_\ell$ and $y_\ell := \widetilde{g}_{\ell+1} - \widetilde{g}_\ell$ and update the subspace by **updateSYH**.
end for

4 Numerical results

In this section, we give a comprehensive comparison of **SSBBO** with several robust and fast solvers for all 568 unconstrained problems from the `CUTEst` [11] collection of test problems for optimization with up to 9000 variables. For all problems, we use the standard initial points given by `CUTEst` collection.

The number of function evaluations and time in milliseconds (excluding the setup time for the objective function) are denoted by `nf` and `msec`, respectively. Let `nfmax` and `secmax` be the limit for `nf` and `sec` (the time in seconds excluding the setup time for the objective function). We choose

$$\texttt{nfmax} \in \begin{cases} \{100n, 500n, 1000n\} & \text{if } 1 \le n \le 30, \\ \{100n, 500n, 1000n\} & \text{if } 31 \le n \le 1000, \\ \{100n, 500n\} & \text{if } 1001 \le n \le 9000, \end{cases}$$

and

$$\texttt{secmax} = \begin{cases} 180 \text{ sec} & \text{if } 1 \le n \le 1000, \\ 480 \text{ sec} & \text{if } 1001 \le n \le 9000. \end{cases}$$

Denote by $\mathcal{S}$ the list of compared solvers and by $\mathcal{P}$ the list of problems. We use the data profile of Moré & Wild [26] and the performance profile of Dolan & Moré [9] to determine the robustness and efficiency of all compared solvers. The data profile

$$\delta_s(\kappa) := \frac{1}{|\mathcal{P}|}\left|\left\{p \in \mathcal{P} \;\middle|\; cr_{p,s} := \frac{c_{p,s}}{n_p + 1} \le \kappa\right\}\right| \tag{43}$$

of the solver s is the fraction of problems that the solver s can solve with κ groups of $n_p + 1$ function evaluations. Here n_p is the dimension of the problem p, $c_{p,s}$ is the **cost measure** of the solver s to solve the problem p and $cr_{p,s}$ is the **cost ratio** of the solver s to solve the problem p. The performance profile

$$\rho_s(\tau) := \frac{1}{|\mathcal{P}|}\left|\left\{p \in \mathcal{P} \;\middle|\; pr_{p,s} := \frac{c_{p,s}}{\min(c_{p,\overline{s}} \mid \overline{s} \in S)} \le \tau\right\}\right| \tag{44}$$

of the solver s is the fraction of problems that the performance ratio $pr_{p,s}$ is at most τ. Note that $\rho_s(1)$ is the fraction of problems that the solver s wins compared to the other solvers, while $\rho_s(\tau)$ ($\delta_s(\kappa)$) is the fraction of problems for sufficiently large τ (κ) that the solver s can solve.

To compare **SSBBO** with known solvers, two cost measures `nf` and `msec` are used. The **efficiency** $e_{p,s}$ of the solver s to solve the problem p is the inverse of the performance ratio $pr_{p,s}$. Efficiency measures the ability of a solver $s \in \mathcal{S}$ compared to an ideal solver. In this paper, the number of function evaluations is taken as a suitable cost measure, and the efficiency with respect to this measure is called the `nf` efficiency.

To determine the convergence speed of the solver s to reach a minimum of the smooth exact function f, we define the quotients

$$q_s := (f_s - f_{\text{opt}})/(f_0 - f_{\text{opt}}) \quad \text{for } s \in \mathcal{S}.$$

Such quotients are not available in real applications. Here, f_s denotes the best function value found by the solver s, f_0 denotes the function value at the starting point (common to all solvers), and f_{opt} denotes the function value at the best point known to the algorithm (in most cases, a global minimizer or at least a better local minimizer) found by performing a sequence of gradient-based and local/global gradient-free solvers; see [20].

We consider a problem solved by the solver s if $q_s \leq 10^{-4}$. Otherwise, the problem is unsolved because either `nfmax` or `secmax` is exceeded.

We compared two versions of **SSBBO** with the best solvers for unconstrained optimization problems from the literature, **UOBYQA** of Powell [30], **BCDFO** of Gratton et al. [14], **BFO** of Porcelli & Toint [29], **NMSMAX** of Higham [16], **NELDER** of Kelley [17], **VRBBO** of Kimiaei & Neumaier [20], and **VRBBON** of Kimiaei [18]. They were selected based on the numerical results in [18,20].

SSBBO1 and **SSBBO2** denote **SSBBO** with memory dimensiosn $m = \min(10, n)$ and $m = \min(20, n)$, respectively. The remaining tuning parameters for are chosen for both versions as:

$$\varepsilon_h = \varepsilon_m, \quad \Delta_{\rm a} = 10^{-8}, \quad \widetilde{\sigma} = 10^{-4}, \quad \widetilde{\rho} = 0.9, \quad \Delta_b = \varepsilon_m, \quad \Delta_\alpha = \varepsilon_m, \quad \gamma_f^1 = 0.5, \quad \gamma_f^2 = 2, \quad \gamma_f^3 = 10^{-12}, \quad \varepsilon_1 = 10^{-8}.$$

• **UOBYQA**, available at

`http://mat.uc.pt/~zhang/software.html`,

is an algorithm that forms quadratic models by interpolation by Powell [30]. Tuning parameters are default.

• **BFO**, available at

`https://github.com/m01marpor/BFO`,

is a trainable stochastic derivative-free solver for mixed integer bound-constrained optimization by Porcelli & Toint [29]. Tuning parameters are default.

• **BCDFO**, obtained from Anke Troeltzsch (personal communication), is a deterministic model-based trust-region algorithm for derivative-free bound-constrained minimization by Gratton et al. [14]. Tuning parameters are default.

• **FMINUNC**, available at the Matlab Optimization Toolbox at

`https://ch.mathworks.com/help/optim/ug/fminunc.html`,

is a deterministic quasi-Newton or trust-region algorithm. **FMINUNC** is used with the following options set by `optimoptions`:

```
opts = optimoptions(@fminunc),'Algorithm','quasi-newton'
'Display', 'Iter','MaxIter',Inf,'MaxFunEvals', limits.nfmax
'TolX', 0,'TolFun',0,'ObjectiveLimit',-1e-50).
```

• **FMINUNC1** is **FMINUNC** using the limited memory BFGS suggested by Liu & Nocedal [24] with the memory $m = 10$. Other tuning parameters are default.

• **FMINUNC2** is **FMINUNC** using the limited memory BFGS suggested by Liu & Nocedal [24] with the memory $m = 20$. Other tuning parameters are default.

• **VRBBO** is an efficient stochastic algorithm by Kimiaei & Neumaier [20]; it can be downloaded from

https://www.mat.univie.ac.at/~neum/software/VRBBO/.

• **VRBBON** is an efficient stochastic algorithm by Kimiaei [18]; it can be downloaded from

https://www.mat.univie.ac.at/~kimiaei/software/VRBBON.

• Two versions of Nelder–Mead simplex method for direct search optimization algorithms are **NELDER** by Kelley [17]

https://ctk.math.ncsu.edu/matlab_darts.html

and **NMSMAX** by Higham, obtained from

http://www.ma.man.ac.uk/~higham/mctoolbox/

For both of them, the default parameters are used.

As described in the introduction, a solver is said to be robust if it has the highest number of solved problems, and efficient if it has the lowest relative cost for function evaluations. A solver is competitive if it is both robust and efficient.

We summarize the numerical results in detail in Subsection 6. **SSBBO** is more robust and efficient than solvers using standard and limited memory quasi-Newton methods not only in low dimensions, but also in medium and high dimensions. In most cases, **SSBBO** is comparable to or even better than well-known model-based solvers in low dimensions in terms of robustness and even efficiency. As a result, **SSBBO** maintains its robustness and efficiency in low to high dimensions compared to the other solvers.

5 Conclusion

An efficient and robust subspace algorithm (**SSBBO**) for unconstrained black box optimization problems in low to high dimensions was introduced. It was able to achieve a significant reduction in the function value by constructing an efficient direction with limited memory along which a successful prediction of a reduction in the model function and its gradient norm could be achieved.

In the presence of the inexact function value and gradient, a worst-case complexity bound on the number of iterations and function evaluations was found for our algorithm, independent of the choice of search directions, which is consistent with that of Berahas et al. [2].

The numerical results confirm that the new subspace method is more efficient and robust than the standard quasi-Newton method and the traditional limited memory, and even comparable to the model-based methods in terms of the efficiency and robustness.

6 Additional material for SSBBO

6.1 Small scale problems

#100 denotes the total number of test problems in which the solver required the least number of `nf`, and !100 denotes the total number of test problems in which the solver was the only one to require so many `nf`. T_{mean} is the mean of the time in seconds taken by a solver to solve the test problems selected from the list of test problems, excluding the time taken for unsolved problems. In all tables,
• the efficiencies are given as percentages, so larger efficiencies indicate better average behaviour and zero efficiency indicates failure.
• all values (against zero) are rounded to whole numbers.
'n' and 't' indicate that `nf` $\geq$ `nfmax` and `sec` $\geq$ `secmax` have been reached, respectively. If the algorithm failed for other reasons, the sign 'f' is used.

Table 1 compares the summary statistics for $1 < n \leq 30$ with the small budget `nfmax` $= 100n$:
• For $1 < n \leq 30$, **SSBBO1** and **SSBBO2** are more robust than others, solve 163 and 162 out of 192 problems, respectively, and have the second highest `nf` efficiency (48%) while **UOBYQA** has the highest `nf` efficiency (49%).
• For $1 < n \leq 2$, **BCDFO**, **SSBBO1**, **SSBBO2**, and **NMSMAX** are more robust than others, while **UOBYQA** has the highest `nf` efficiency (72%).
• For $3 \leq n \leq 5$, **SSBBO1**, **SSBBO2**, and **UOBYQA** are more robust than others and solve 35 problems out of 39 problems. **UOBYQA**, **BCDFO**, and **SSBBO1** have the highest, second highest, and third highest `nf` efficiency, respectively.
• For $6 \leq n \leq 10$, **UOBYQA** and **SSBBO1** (**SSBBO2**) are more robust than others, solving 58 out of 76 problems, while **SSBBO1** (**SSBBO2**) has the highest `nf` efficiency (48%).
• For $11 \leq n \leq 30$, **SSBBO1** and (**SSBBO2**, **FMINUNC**, and **VRBBO**) are more robust than others, solving 40 and 39 out of 43 problems, while **VRBBON**, **SSBBO1**, and **SSBBO2** have the highest, second highest, and third highest `nf` efficiency (54%, 52%, and 51%), respectively.

Table 1: Results for $1 \leq n \leq 30$ with $\texttt{nfmax} = 100n$

stopping test:		$q_f \leq$ 0.0001,		sec $\leq$ 180,		nf $\leq$ 100*n				
178 of 192 problems solved										mean efficiency in %
dim∈[1,30]						# of anomalies				for cost measure
solver		solved	#100	!100	τ_{mean}	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	163	30	0	79	29	0	0	48	40
SSBBO2	**ssbb2**	162	33	3	75	30	0	0	48	39
UOBYQA	**uob**	159	32	22	386	30	0	3	49	47
FMINUNC	**func**	145	12	8	69	16	0	31	39	41
NMSMAX	**nmsm**	144	4	2	95	47	0	1	26	39
FMINUNC1	**func1**	140	14	0	51	47	0	5	39	42
FMINUNC2	**func2**	140	14	0	51	47	0	5	40	42
BCDFO	**bcd**	135	41	36	2961	0	10	47	40	12
VRBBO	**vrbb**	134	18	9	150	58	0	0	28	22
VRBBON	**vrbbn**	131	37	33	268	61	0	0	34	15
BFO	**bfo**	112	6	4	141	0	0	80	18	11
NELDER	**neld**	101	4	4	131	82	0	9	15	19
33 of 34 problems without bounds solved										mean efficiency in %
dim∈[1,2]						# of anomalies				for cost measure
solver		solved	#100	!100	τ_{mean}	#n	#t	#f	nf	msec
BCDFO	**bcd**	31	10	8	63	0	0	3	67	24
SSBBO1	**ssbb1**	30	1	0	44	4	0	0	41	19
SSBBO2	**ssbb2**	30	1	0	43	4	0	0	41	21
NMSMAX	**nmsm**	30	3	2	17	4	0	0	43	62
UOBYQA	**uob**	29	12	9	16	5	0	0	72	69
NELDER	**neld**	28	2	2	21	3	0	3	34	39
FMINUNC1	**func1**	28	2	0	32	6	0	0	43	25
FMINUNC2	**func2**	28	2	0	33	6	0	0	43	27
VRBBON	**vrbbn**	27	6	4	71	7	0	0	39	11
BFO	**bfo**	26	1	0	87	0	0	8	29	6
VRBBO	**vrbb**	26	3	1	40	8	0	0	36	20
FMINUNC	**func**	25	1	0	27	5	0	4	39	25
36 of 39 problems without bounds solved										mean efficiency in %
dim∈[3,5]						# of anomalies				for cost measure
solver		solved	#100	!100	τ_{mean}	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	35	4	0	70	4	0	0	51	35
SSBBO2	**ssbb2**	35	4	0	69	4	0	0	51	35
UOBYQA	**uob**	35	7	5	49	3	0	1	63	65
NMSMAX	**nmsm**	33	1	0	37	5	0	1	42	66
NELDER	**neld**	33	1	1	65	5	0	1	29	37
BCDFO	**bcd**	32	14	12	109	0	0	7	60	23
FMINUNC	**func**	32	3	2	51	2	0	5	48	47
FMINUNC1	**func1**	29	1	0	53	10	0	0	36	33
FMINUNC2	**func2**	29	1	0	52	10	0	0	36	34
BFO	**bfo**	26	3	2	113	0	0	13	30	14
VRBBON	**vrbbn**	26	6	4	158	13	0	0	29	18
VRBBO	**vrbb**	23	2	1	79	16	0	0	22	16
67 of 76 problems without bounds solved										mean efficiency in %
dim∈[6,10]						# of anomalies				for cost measure
solver		solved	#100	!100	τ_{mean}	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	58	19	0	82	18	0	0	48	41
SSBBO2	**ssbb2**	58	19	0	84	18	0	0	48	40
UOBYQA	**uob**	58	9	6	147	16	0	2	43	45
BCDFO	**bcd**	49	15	14	2399	0	0	27	34	7
FMINUNC	**func**	49	5	4	73	6	0	21	34	42
NMSMAX	**nmsm**	48	0	0	102	28	0	0	17	26
FMINUNC1	**func1**	47	4	0	52	25	0	4	35	42
FMINUNC2	**func2**	47	4	0	53	25	0	4	35	40
VRBBO	**vrbb**	46	9	5	166	30	0	0	23	22
VRBBON	**vrbbn**	43	9	9	354	33	0	0	24	12
BFO	**bfo**	38	2	2	147	0	0	38	13	14
NELDER	**neld**	34	1	1	213	37	0	5	7	9
42 of 43 problems without bounds solved										mean efficiency in %
dim∈[11,30]						# of anomalies				for cost measure
solver		solved	#100	!100	τ_{mean}	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	40	6	0	108	3	0	0	52	58
SSBBO2	**ssbb2**	39	9	3	92	4	0	0	51	56
FMINUNC	**func**	39	3	2	106	3	0	1	37	49
VRBBO	**vrbb**	39	4	2	245	4	0	0	35	28
UOBYQA	**uob**	37	4	2	1371	6	0	0	27	19
FMINUNC1	**func1**	36	7	0	63	6	0	1	46	63
FMINUNC2	**func2**	36	7	0	61	6	0	1	47	64
VRBBON	**vrbbn**	35	16	16	395	8	0	0	54	21
NMSMAX	**nmsm**	33	0	0	212	10	0	0	15	20
BCDFO	**bcd**	23	2	2	12035	0	10	10	13	0
BFO	**bfo**	22	0	0	226	0	0	21	7	8
NELDER	**neld**	6	0	0	538	37	0	0	2	2

The results obtained for $1 < n \leq 30$ with the medium budget $\texttt{nfmax} = 500n$ are summarized in Table 2:

• For $1 < n \leq 30$, **SSBBO1**, **SSBBO2** and **UOBYQA** are more robust than others, solving 175 out of 192 problems, while **UOBYQA**, **SSBBO1** (**SSBBO2**) have the highest and second highest `nf` efficiency (52% and 49%), respectively.

• For $1 < n \leq 2$, **UOBYQA** (**BCDFO**) and **SSBBO1** (**SSBBO2**) are more robust than others, solving 34 and 33 out of 34 problems, respectively, and **UOBYQA**, **BCDFO** and **NMSMAX** have the highest, second highest, and third highest `nf` efficiency (77%, 70%, and 46%), respectively.

• For $3 \leq n \leq 5$, **SSBBO1** (**SSBBO2**) are more robust than others and solve 36 out of 39 problems,

and **UOBYQA**, **BCDFO**, and **SSBBO1** (**SSBBO2**) have the highest, second highest, and third highest `nf` (63%, 60%, and 50%), respectively.

• For $6 \leq n \leq 10$, **UOBYQA** and **SSBBO1** (**SSBBO2**) are more robust than others, solving 65 and 64 out of 76 problems, respectively, while **SSBBO1** (**SSBBO2**) and **UOBYQA** have the highest and second highest `nf` efficiency (50% and 48%), respectively.

• For $11 \leq n \leq 30$, **SSBBO1** (**SSBBO2**, **FMINUNC**, and **VRBBO**) is more robust than others and solves 42 out of 43 problems, while **SSBBO1** (**SSBBO2**) and **VRBBON** have the highest and second highest `nf` efficiency (56% and 52%), respectively.

Table 2: Results for $1 < n \leq 30$ with $\texttt{nfmax} = 500n$

stopping test:		$q_f \leq$ 0.0001,			sec $\leq$ 180,			nf $\leq$ 500*n			
183 of 192 problems solved										mean efficiency in %	
dim∈(1,30]						# of anomalies			for cost measure		
solver		solved	#100	!100	$\mathcal{T}$mean	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	175	29	0	103	17	0	0	49	42
SSBBO2	**ssbb2**	175	32	3	101	17	0	0	49	44
UOBYQA	**uob**	175	34	25	715	9	0	8	52	49
VRBBO	**vrbb**	167	20	13	276	25	0	0	32	23
FMINUNC1	**func1**	164	14	0	122	19	0	9	41	43
FMINUNC2	**func2**	164	14	0	121	19	0	9	41	42
VRBBON	**vrbbn**	164	37	33	644	28	0	0	37	18
NMSMAX	**nmsm**	160	4	2	181	27	0	5	27	40
FMINUNC	**func**	158	13	9	130	1	0	33	40	42
BCDFO	**bcd**	156	40	35	5873	0	12	24	42	13
NELDER	**neld**	148	4	4	593	21	0	23	17	20
BFO	**bfo**	143	6	4	251	0	0	49	19	18
34 of 34 problems without bounds solved										mean efficiency in %	
dim∈(1,2]						# of anomalies			for cost measure		
solver		solved	#100	!100	$\mathcal{T}$mean	#n	#t	#f	nf	msec
BCDFO	**bcd**	34	10	8	102	0	0	0	70	31
UOBYQA	**uob**	34	13	10	52	0	0	0	77	76
SSBBO1	**ssbb1**	33	1	0	61	1	0	0	44	20
SSBBO2	**ssbb2**	33	1	0	59	1	0	0	44	21
NMSMAX	**nmsm**	32	3	2	25	1	0	1	46	58
FMINUNC1	**func1**	32	2	0	53	2	0	0	45	26
FMINUNC2	**func2**	32	2	0	56	2	0	0	45	21
BFO	**bfo**	31	1	0	106	0	0	3	32	16
NELDER	**neld**	31	2	2	38	0	0	3	37	38
VRBBON	**vrbbn**	31	6	4	81	3	0	0	42	21
FMINUNC	**func**	30	1	0	56	0	0	4	44	25
VRBBO	**vrbb**	30	3	1	64	4	0	0	40	19
36 of 39 problems without bounds solved										mean efficiency in %	
dim∈[3,5]						# of anomalies			for cost measure		
solver		solved	#100	!100	$\mathcal{T}$mean	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	36	4	0	70	3	0	0	50	39
SSBBO2	**ssbb2**	36	4	0	70	3	0	0	50	42
NELDER	**neld**	35	1	1	80	2	0	2	30	40
BCDFO	**bcd**	35	13	11	158	0	0	4	60	24
UOBYQA	**uob**	35	7	5	53	2	0	2	63	64
VRBBON	**vrbbn**	35	6	4	264	4	0	0	32	17
NMSMAX	**nmsm**	34	1	0	43	3	0	2	42	68
FMINUNC1	**func1**	34	1	0	96	5	0	0	37	37
FMINUNC2	**func2**	34	1	0	99	5	0	0	37	35
VRBBO	**vrbb**	34	3	2	149	5	0	0	30	24
FMINUNC	**func**	33	3	2	60	1	0	5	47	48
BFO	**bfo**	30	3	2	114	0	0	9	30	28
71 of 76 problems without bounds solved										mean efficiency in %	
dim∈[6,10]						# of anomalies			for cost measure		
solver		solved	#100	!100	$\mathcal{T}$mean	#n	#t	#f	nf	msec
UOBYQA	**uob**	65	10	8	355	7	0	4	48	46
SSBBO1	**ssbb1**	64	18	0	109	12	0	0	50	44
SSBBO2	**ssbb2**	64	18	0	107	12	0	0	50	47
VRBBO	**vrbb**	61	10	8	366	15	0	0	26	24
NMSMAX	**nmsm**	59	0	0	256	15	0	2	18	28
BCDFO	**bcd**	59	15	14	4131	0	0	17	36	7
NELDER	**neld**	58	1	1	424	2	0	16	10	10
FMINUNC1	**func1**	58	4	0	149	10	0	8	37	42
FMINUNC2	**func2**	58	4	0	147	10	0	8	37	42
VRBBON	**vrbbn**	58	9	9	799	18	0	0	26	14
FMINUNC	**func**	53	6	5	149	0	0	23	36	42
BFO	**bfo**	47	2	2	242	0	0	29	14	17
42 of 43 problems without bounds solved										mean efficiency in %	
dim∈[11,30]						# of anomalies			for cost measure		
solver		solved	#100	!100	$\mathcal{T}$mean	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	42	6	0	154	1	0	0	52	59
SSBBO2	**ssbb2**	42	9	3	150	1	0	0	52	57
FMINUNC	**func**	42	3	2	215	0	0	1	38	49
VRBBO	**vrbb**	42	4	2	399	1	0	0	36	24
UOBYQA	**uob**	41	4	2	2402	0	0	2	28	19
FMINUNC1	**func1**	40	7	0	159	2	0	1	48	63
FMINUNC2	**func2**	40	7	0	153	2	0	1	49	66
VRBBON	**vrbbn**	40	16	16	1190	3	0	0	56	21
BFO	**bfo**	35	0	0	511	0	0	8	9	11
NMSMAX	**nmsm**	35	0	0	332	8	0	0	15	21
BCDFO	**bcd**	28	2	2	23698	0	12	3	14	0
NELDER	**neld**	24	0	0	2464	17	0	2	4	4

Table 3 provides the results for $1 < n \leq 30$ with the large budget $\texttt{nfmax} = 1000n$:
• For $1 < n \leq 30$, **SSBBO1** (**SSBBO2**) and **UOBYQA** are more robust than others, solving 178 and 176 out of 192 problems, respectively, while **UOBYQA** and **SSBBO1** (**SSBBO2**) have the highest and second highest `nf` efficiency (52% and 50%), respectively.
• For $1 < n \leq 2$, **SSBBO1** (**SSBBO2**, **UOBYQA**, and **BCDFO**) is more robust than others and solves 34 out of 34 problems, while **UOBYQA**, **BCDFO**, and **NMSMAX** have the highest, second highest, and third highest `nf` efficiency (77%, 71%, and 46%), respectively.
• For $3 \leq n \leq 5$, **SSBBO1** (**SSBBO2**, **FMINUNC1**, and **FMINUNC2**) is more robust than others and solves 36 out of 39 problems, while **UOBYQA**, **BCDFO**, and **SSBBO1** (**SSBBO2**) have the highest, second highest, and third highest `nf` (63%, 61%, and 51%), respectively.
• For $6 \leq n \leq 10$, **SSBBO1** (**SSBBO2**), **UOBYQA**, and **VRBBO** are more robust than others, solving 66 out of 76 problems, while **SSBBO1** (**SSBBO2**) and **UOBYQA** have the highest and second highest `nf` efficiency (51% and 48%), respectively.
• For $11 \leq n \leq 30$, **SSBBO1** (**SSBBO2**, **FMINUNC1**, **VRBBO**, and **FMINUNC2**) is more robust than others and solves 42 out of 43 problems, while **SSBBO1** (**SSBBO2**) and **VRBBON** have the highest and second highest `nf` efficiency (56% and 52%), respectively.

Table 3: Results for $1 < n \leq 30$ with $\texttt{nfmax} = 1000n$

stopping test:		$q_f \leq$ 0.0001,			sec $\leq$ 180,		nf $\leq$ 1000*n			
184 of 192 problems solved										mean efficiency in %
dim∈(1,30]						# of anomalies				for cost measure
solver		solved	#100	!100	$\mathcal{T}$mean	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	178	29	0	127	14	0	0	50	44
SSBBO2	**ssbb2**	178	32	3	125	14	0	0	50	45
UOBYQA	**uob**	176	35	26	762	7	0	9	52	51
VRBBO	**vrbb**	172	19	12	299	20	0	0	32	23
FMINUNC1	**func1**	171	14	0	221	2	0	19	41	44
FMINUNC2	**func2**	170	14	0	207	3	0	19	41	47
VRBBON	**vrbbn**	166	37	33	676	26	0	0	37	16
NMSMAX	**nmsm**	162	4	2	214	24	0	6	28	41
FMINUNC	**func**	158	13	9	133	1	0	33	40	45
BCDFO	**bcd**	157	41	36	6907	0	16	19	43	13
NELDER	**neld**	156	4	4	850	8	0	28	18	22
BFO	**bfo**	147	6	4	386	0	0	45	20	21
34 of 34 problems without bounds solved										mean efficiency in %
dim∈(1,2]						# of anomalies				for cost measure
solver		solved	#100	!100	$\mathcal{T}$mean	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	34	1	0	77	0	0	0	45	27
SSBBO2	**ssbb2**	34	1	0	76	0	0	0	45	27
BCDFO	**bcd**	34	10	8	96	0	0	0	71	25
UOBYQA	**uob**	34	13	10	46	0	0	0	77	77
NMSMAX	**nmsm**	32	3	2	27	1	0	1	46	68
FMINUNC1	**func1**	32	2	0	56	1	0	1	45	26
FMINUNC2	**func2**	32	2	0	54	1	0	1	45	33
BFO	**bfo**	31	1	0	90	0	0	3	32	27
NELDER	**neld**	31	2	2	35	0	0	3	37	49
VRBBO	**vrbb**	31	3	1	96	3	0	0	37	17
VRBBON	**vrbbn**	31	6	4	102	3	0	0	42	16
FMINUNC	**func**	30	1	0	54	0	0	4	44	37
36 of 39 problems without bounds solved										mean efficiency in %
dim∈[3,5]						# of anomalies				for cost measure
solver		solved	#100	!100	$\mathcal{T}$mean	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	36	4	0	74	3	0	0	51	38
SSBBO2	**ssbb2**	36	4	0	71	3	0	0	51	41
FMINUNC1	**func1**	36	1	0	135	0	0	3	38	37
FMINUNC2	**func2**	36	1	0	138	0	0	3	38	39
NELDER	**neld**	35	1	1	78	2	0	2	30	40
BCDFO	**bcd**	35	14	12	158	0	0	4	61	26
UOBYQA	**uob**	35	7	5	53	2	0	2	63	69
VRBBON	**vrbbn**	35	6	4	267	4	0	0	32	16
NMSMAX	**nmsm**	34	1	0	45	3	0	2	43	62
FMINUNC	**func**	33	3	2	60	1	0	5	48	48
VRBBO	**vrbb**	33	2	1	148	6	0	0	26	20
BFO	**bfo**	30	3	2	108	0	0	9	31	31
72 of 76 problems without bounds solved										mean efficiency in %
dim∈[6,10]						# of anomalies				for cost measure
solver		solved	#100	!100	$\mathcal{T}$mean	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	66	18	0	160	10	0	0	51	45
SSBBO2	**ssbb2**	66	18	0	161	10	0	0	51	47
UOBYQA	**uob**	66	11	9	497	5	0	5	48	48
VRBBO	**vrbb**	66	10	8	418	10	0	0	30	25
NMSMAX	**nmsm**	61	0	0	338	12	0	3	19	29
FMINUNC1	**func1**	61	4	0	295	1	0	14	37	43
FMINUNC2	**func2**	61	4	0	296	1	0	14	37	46
BCDFO	**bcd**	60	15	14	6321	0	2	14	37	8
VRBBON	**vrbbn**	60	9	9	875	16	0	0	27	14
NELDER	**neld**	58	1	1	437	0	0	18	10	11
FMINUNC	**func**	53	6	5	156	0	0	23	36	43
BFO	**bfo**	48	2	2	287	0	0	28	14	17
42 of 43 problems without bounds solved										mean efficiency in %
dim∈[11,30]						# of anomalies				for cost measure
solver		solved	#100	!100	$\mathcal{T}$mean	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	42	6	0	159	1	0	0	52	59
SSBBO2	**ssbb2**	42	9	3	155	1	0	0	52	59
FMINUNC	**func**	42	3	2	218	0	0	1	38	52
FMINUNC1	**func1**	42	7	0	315	0	0	1	49	66
VRBBO	**vrbb**	42	4	2	380	1	0	0	36	28
UOBYQA	**uob**	41	4	2	2387	0	0	2	28	19
FMINUNC2	**func2**	41	7	0	254	1	0	1	49	68
VRBBON	**vrbbn**	40	16	16	1180	3	0	0	56	21
BFO	**bfo**	38	0	0	971	0	0	5	9	13
NMSMAX	**nmsm**	35	0	0	332	8	0	0	15	21
NELDER	**neld**	32	0	0	3231	6	0	5	4	3
BCDFO	**bcd**	28	2	2	24869	0	14	1	14	1

From performance and data profiles of Figures 1, we conclude that **SSBBO1** (**SSBBO2**) and **UOBYQA** are more robust than others, respectively, while **UOBYQA** and **SSBBO1** (**SSBBO2**) have more efficient than others, respectively.

What is interesting about Tables 1-3 and Figures 1 is that **SSBBO1** and **SSBBO2** are more efficient and robust than **FMINUNC** using the standard BFGS method and **FMINUNC1** and **FMINUNC2** using the limited memory BFGS method in terms of the number of problems solved and the `nf` efficiency, especially at low and medium budgets.

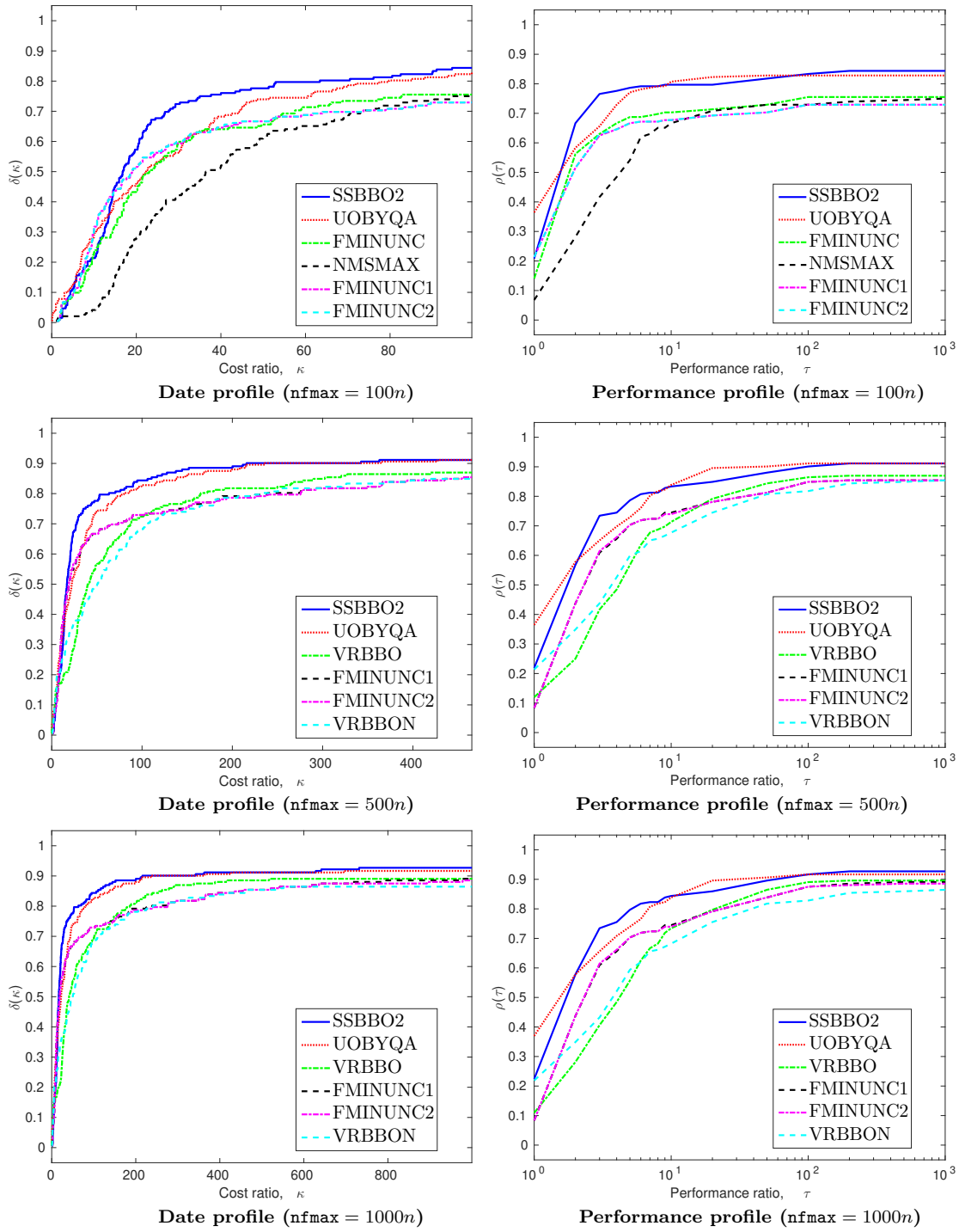


Fig. 1: Results for small scale problems ($n \leq 30$). Left: Data profile $\delta(\kappa)$ in dependence of a bound κ on the cost ratio, see (43). Right: Performance profile $\rho(\tau)$ in dependence of a bound τ on the performance ratio, see (44). Problems solved by no solver are ignored.

6.2 Medium scale problems

Table 4 shows the results for $30 < n \leq 1000$ with the small budget `nfmax` $= 100n$. **SSBBO2** and **SSBBO1** are more robust than others, solving 212 and 210 out of 267 problems, respectively, while **SSBBO2** and **SSBBO1** have the highest and second highest `nf` efficiency (58% and 56%), respectively.

Table 4: Results for $30 < n \leq 1000$ with `nfmax` $= 100n$

stopping test:		$q_f \leq$ 0.0001,		sec $\leq$ 180,		nf $\leq$ 100*n				
227 of 267 problems without bounds solved									mean efficiency in %	
dim∈(30,1000]						# of anomalies			for cost measure	
solver		solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
SSBBO2	**ssbb2**	212	82	14	840	54	1	0	58	62
SSBBO1	**ssbb1**	209	76	9	809	57	1	0	56	61
FMINUNC	**func**	191	37	20	1670	33	0	43	43	33
VRBBO	**vrbb**	190	17	15	5080	75	2	0	31	20
FMINUNC1	**func1**	188	42	1	1013	68	1	10	46	53
FMINUNC2	**func2**	188	43	1	870	69	1	9	46	53
VRBBON	**vrbbn**	174	78	71	7737	89	4	0	43	14

Our results, for $30 < n \leq 1000$ with the medium budget `nfmax` $= 500n$, are shown in Table 5. **SSBBO2** and **SSBBO1** are more robust than others, solving 226 and 223 out of 267 problems, respectively, while **SSBBO2** and **SSBBO1** have the highest and second highest `nf` efficiency (60% and 59%), respectively.

Table 5: Results for $30 < n \leq 1000$ with `nfmax` $= 500n$

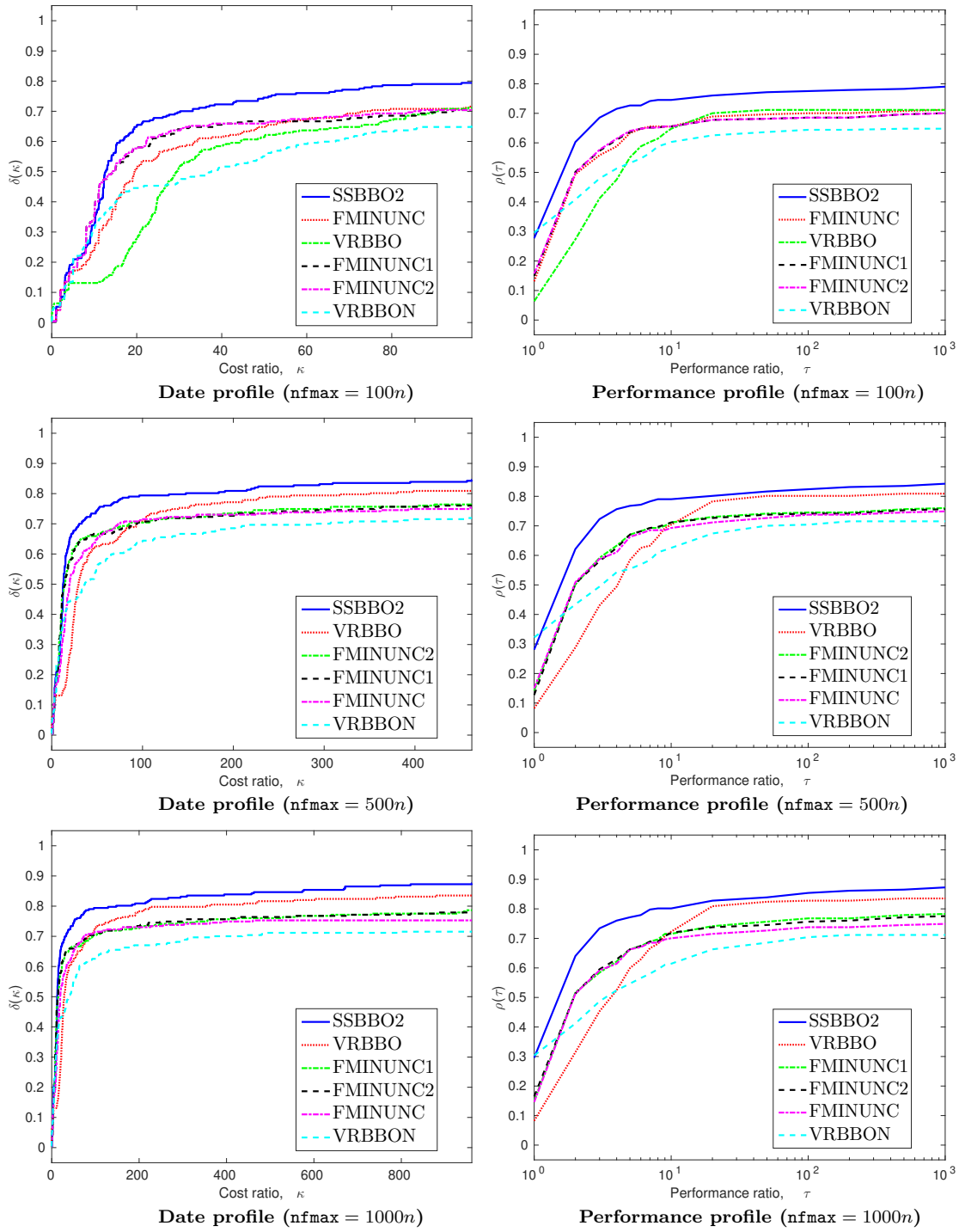
stopping test:		$q_f \leq$ 0.0001,		sec $\leq$ 180,		nf $\leq$ 500*n				
244 of 267 problems without bounds solved									mean efficiency in %	
dim∈(30,1000]						# of anomalies			for cost measure	
solver		solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
SSBBO2	**ssbb2**	226	83	16	1679	40	1	0	60	66
SSBBO1	**ssbb1**	223	77	11	1042	43	1	0	59	66
VRBBO	**vrbb**	216	22	21	9902	32	19	0	34	19
FMINUNC2	**func2**	204	37	1	2307	50	1	12	47	56
FMINUNC1	**func1**	203	36	1	2420	51	1	12	47	55
FMINUNC	**func**	201	41	24	1997	14	1	51	45	37
VRBBON	**vrbbn**	192	86	81	7926	45	30	0	45	16

The results, shown in Table 6 for $30 < n \leq 1000$ with the large budget `nfmax` $= 1000n$, show that **SSBBO2** and **SSBBO1** are more robust than others, solving 234 and 229 out of 267 problems, respectively, while **SSBBO2** and **SSBBO1** have the highest and second highest `nf` efficiency (62% and 60%), respectively.

Table 6: Results for $30 < n \leq 1000$ with `nfmax` $= 1000n$

stopping test:		$q_f \leq$ 0.0001,		sec $\leq$ 180,		nf $\leq$ 1000*n				
246 of 267 problems without bounds solved									mean efficiency in %	
dim∈(30,1000]						# of anomalies			for cost measure	
solver		solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
SSBBO2	**ssbb2**	234	87	19	3143	25	8	0	62	64
SSBBO1	**ssbb1**	229	80	13	2842	32	6	0	60	65
VRBBO	**vrbb**	223	21	20	10123	23	21	0	35	20
FMINUNC1	**func1**	210	42	1	3758	27	1	29	47	56
FMINUNC2	**func2**	208	43	1	3547	38	2	19	48	56
FMINUNC	**func**	201	40	23	1953	7	5	54	45	37
VRBBON	**vrbbn**	191	80	74	8460	38	38	0	44	15

From performance and data profiles of Figures 2, we conclude that **SSBBO1** (**SSBBO2**) is more robust and efficient than others, e.g., solvers using standard and limited memory quasi-Newton methods.



Date profile (`nfmax` $= 100n$) **Performance profile** (`nfmax` $= 100n$)

Date profile (`nfmax` $= 500n$) **Performance profile** (`nfmax` $= 500n$)

Date profile (`nfmax` $= 1000n$) **Performance profile** (`nfmax` $= 1000n$)

Fig. 2: Results for medium scale problems ($30 < n \leq 1000$). Details as in Figure 1.

6.3 Large scale problems

The results, for $1000 < n \leq 9000$ with the small budget `nfmax` $= 100n$, are summarized in Table 7 with the goal of identifying the competitive solver. **SSBBO2** is more robust than others, solving 87 out of 109 problems, while **VRBBON** has the highest highest `nf` efficiency.

Table 7: Results for $1000 < n \leq 9000$ with `nfmax` $= 100n$

99 of 109 problems without bounds solved									mean efficiency in %	
dim∈(1000,9000]						# of anomalies			for cost measure	
solver		solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
SSBBO2	**ssbb2**	87	18	2	33685	17	5	0	48	58
VRBBO	**vrbb**	86	5	5	116710	6	17	0	31	21
SSBBO1	**ssbb1**	85	17	1	34749	20	4	0	48	58
FMINUNC1	**func1**	83	17	1	28944	21	5	0	53	67
FMINUNC	**func**	83	15	7	61249	12	5	9	44	35
FMINUNC2	**func2**	82	16	0	25069	22	5	0	52	67
VRBBON	**vrbbn**	80	59	58	85530	7	22	0	64	24

The results, for $1000 < n \leq 9000$ with the medium budget `nfmax` $= 500n$, are summarized in Table 7 with the goal of identifying the competitive solver. **SSBBO2** (**STBBO1**) is more robust than others, solving 90 out of 109 problems, while **VRBBON** has the highest highest `nf` efficiency.

Table 8: Results for $1000 < n \leq 9000$ with `nfmax` $= 500n$

stopping test:		$q_f \leq$ `0.0001,`		`sec` $\leq$ `180,`		`nf` $\leq$ `500*n`				
99 of 109 problems without bounds solved									mean efficiency in %	
dim∈(1000,9000]						# of anomalies			for cost measure	
solver		solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
SSBBO1	**ssbb1**	90	17	1	34796	1	18	0	50	61
SSBBO2	**ssbb2**	90	18	2	33479	1	18	0	50	60
VRBBO	**vrbb**	88	4	4	110178	0	21	0	33	22
FMINUNC1	**func1**	86	19	1	30037	3	20	0	55	67
FMINUNC2	**func2**	86	18	0	29237	3	20	0	55	68
FMINUNC	**func**	85	16	7	60847	2	13	9	46	36
VRBBON	**vrbbn**	80	59	56	81908	0	29	0	64	23

From performance and data profiles of Figures 3, we conclude that **SSBBO2** is more robust than others, while **VRBBON** is more efficient than others. Moreover, **SSBBO2** is robust and efficient than solvers using standard and limited memory quasi-Newton methods.

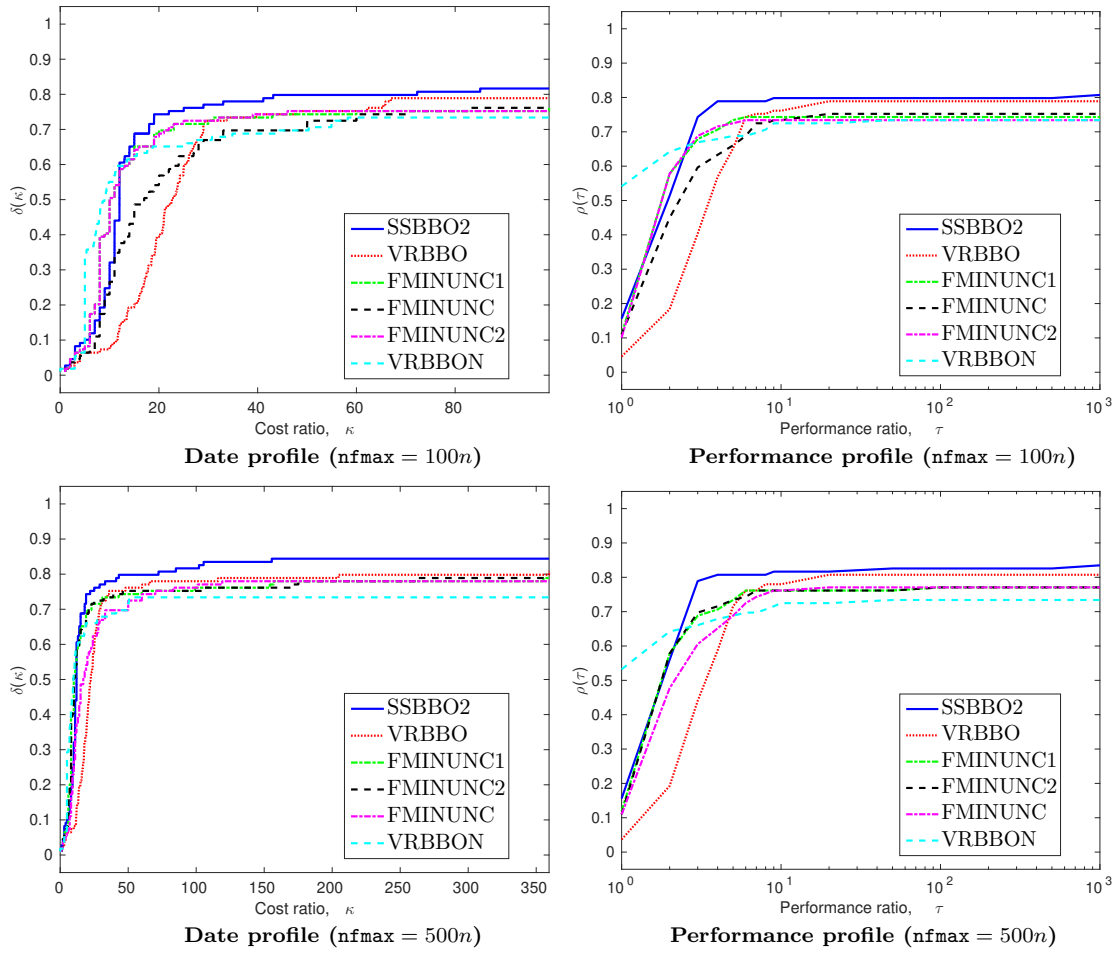


Fig. 3: Results for large scale problems ($1000 < n \leq 9000$). Details as in Figure 1.

References

1. M. Al-Baali and R. Fletcher. An efficient line search for nonlinear least squares. *J. Optim. Theory Appl.* **48** (March 1986), 359–377.
2. A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Found. Comut. Math.* (2021) https://doi.org/10.1007/s10208-021-09513-z.
3. Albert S. Berahas, Richard H. Byrd, and Jorge Nocedal. Derivative-free optimization of noisy functions via quasi-newton methods. *SIAM J. Optim.* **29** (January 2019), 965–993.
4. Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Math. Program.* **63** (January 1994), 129–156.
5. C. Cartis, Ph.R. Sampaio, and Ph.L. Toint. Worst-case evaluation complexity of non-monotone gradient-related algorithms for unconstrained optimization. *Optimization* **64** (January 2014), 1349–1361.
6. A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics (January 2009).
7. F. E. Curtis, Z. Lubberts, and D. P. Robinson. Concise complexity analyses for trust region methods. *Optim. Lett.* **12** (June 2018), 1713–1724.
8. F. E. Curtis, D. P. Robinson, and M. Samadi. A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\varepsilon^{-3/2})$ for nonconvex optimization. *Math. Program.* **162** (May 2016), 1–32.
9. E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.* **91** (January 2002), 201–213.
10. R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, Ltd (May 2000).
11. Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.* **60** (August 2014), 545–557.
12. G. N. Grapiglia, J. Yuan, and Y. Yuan. Nonlinear stepsize control algorithms: Complexity bounds for first- and second order optimality. *J. Optim. Theory Appl.* **171** (September 2016), 980–997.
13. S. Gratton, C. W. Royer, and L. N. Vicente. A decoupled first/second-order steps technique for nonconvex nonlinear unconstrained optimization with improved complexity bounds. *Math. Program.* **179** (September 2018), 195–222.
14. S. Gratton, Ph. L. Toint, and A. Tröltzsch. An active-set trust-region method for derivative-free nonlinear bound-constrained optimization. *Optim. Methods Softw.* **26** (October 2011), 873–894.
15. R. W. Hamming. *Introduction to Applied Numerical Analysis*. Taylor & Francis/Hemisphere, USA (1989).
16. N. J. Higham. Optimization by direct search in matrix computations. *SIAM J. Matrix Anal. Appl.* **14** (April 1993), 317–333.
17. C. T. Kelley. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics (January 1999).
18. M. Kimiaei. Line search in noisy unconstrained black box optimization. `http://www.optimization-online.org/DB_HTML/2020/09/8007.html` (Sep 2020).
19. M. Kimiaei and A. Neumaier. Testing and tuning optimization algorithm. Preprint, Vienna University, Fakultät für Mathematik, Universität Wien, Oskar-Morgenstern-Platz 1, A-1090 Wien, Austria (2019).
20. M. Kimiaei and A. Neumaier. Efficient global unconstrained black box optimization. `http://www.optimization-online.org/DB_HTML/2018/08/6783.html` (Jul 2020).
21. M. Kimiaei, A. Neumaier, and B. Azmi. LMBOPT – a limited memory method for bound-constrained optimization. `http://www.optimization-online.org/DB_HTML/2020/11/8089.html` (Nov 2020).
22. J. Larson, M. Menickelly, and S. M. Wild. Derivative-free optimization methods. *Acta Numer.* **28** (May 2019), 287–404.
23. D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.* **45** (August 1989), 503–528.
24. D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.* **45** (August 1989), 503–528.
25. J. L. Morales and J. Nocedal. Remark on "algorithm 778: L-BFGS-b: Fortran subroutines for large-scale bound constrained optimization". *ACM Trans. Math. Softw.* **38** (November 2011), 1–4.
26. J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.* **20** (January 2009), 172–191.

27. J. Nocedal. Updating quasi-newton matrices with limited storage. *Math Comput.* **35** (September 1980), 773–773.
28. J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer New York (2006).
29. M. Porcelli and P. Toint. Global and local information in structured derivative free optimization with BFO. *arXiv: Optimization and Control* (2020).
30. M. J. D. Powell. UOBYQA: unconstrained optimization by quadratic approximation. *Math. Program.* **92** (May 2002), 555–582.
31. L. M. Rios and N. V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *J. Glob. Optim.* **56** (July 2012), 1247–1293.
32. Y. Xie, R. H. Byrd, and J. Nocedal. Analysis of the BFGS method with errors. *SIAM J. Optim.* **30** (January 2020), 182–209.